NEW

120+ mins of free video

Linux & Open Source Annual

Everything you need to master open source software and operating systems

645 expert tips & tricks

Welcome to

Linux & Open Source Annual

Linux and open source software is, by its very nature, always changing and evolving. By constantly updating and improving, it pushes the boundaries of what can be achieved with software. Whether this takes the form of updates to the many operating systems out there, awesome hacks that make life or computing easier than ever, or new hardware like the Raspberry Pi 2, there's always something groundbreaking on the horizon. We have crammed as much of the year's exciting new information as we possibly can into this annual, from the key developments in the wide-ranging world of open source software, to the many practical tutorials that can help improve your systems and solve network issues. This collection truly is a treasure-trove of Linux and open source knowledge, ready for you to jump right into. So, what are you waiting for?

Linux & Open Source Annual

Imagine Publishing Ltd
Richmond House
33 Richmond Hill
Bournemouth
Dorset BH2 6EZ
☎ +44 (0) 1202 586200
Website: www.imagine-publishing.co.uk
Twitter: @Books_Imagine
Facebook: www.facebook.com/ImagineBookazines

Publishing Director
Aaron Asadi

Head of Design
Ross Andrews

Production Editor
Alex Hoskins

Senior Art Editor
Greg Whitaker

Designer
Abbi Castle

Printed by
William Gibbons, 26 Planetary Road, Willenhall, West Midlands, WV13 3XT

Distributed in the UK, Eire & the Rest of the World by
Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU.
Tel 0203 787 9060 www.marketforce.co.uk

Distributed in Australia by
Network Services (a division of Bauer Media Group), Level 21 Civic Tower, 66-68 Goulburn Street, Sydney, New South Wales 2000, Australia Tel +61 2 8667 5288

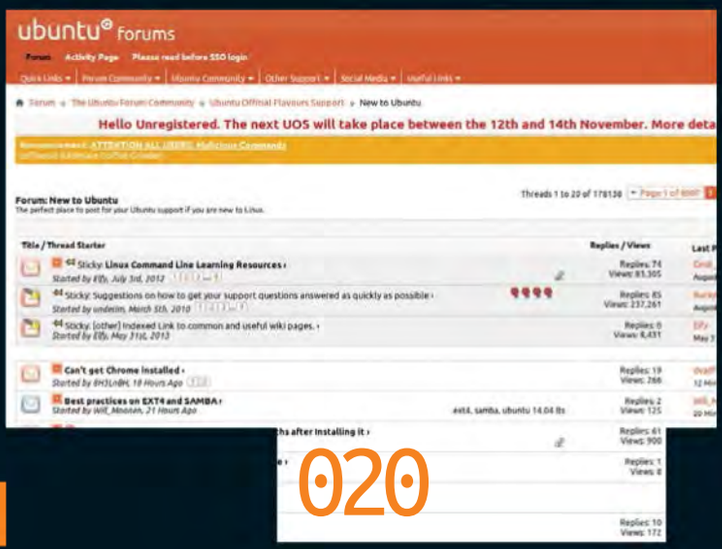
ISBN 978 1785 461 835

Part of the LinuxUser & Developer bookazine series

IMAGINE PUBLISHING

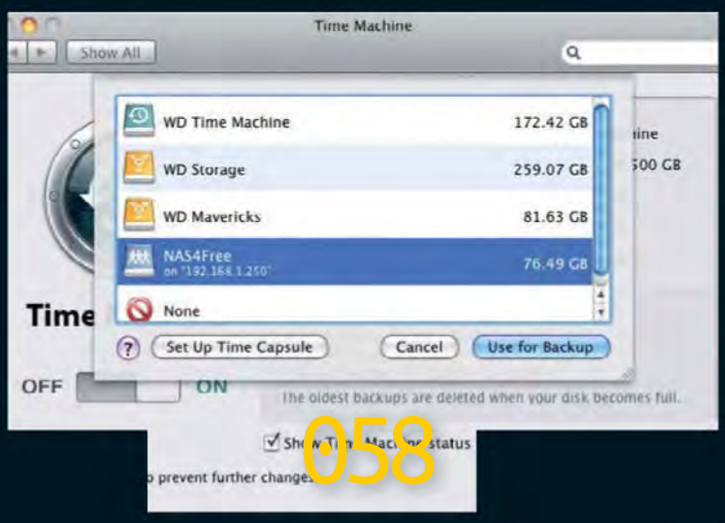
Contents

008 Best free software
All the software you need at home



Open source world

020 Evolution of Ubuntu
028 Get started with Ubuntu Phone
032 Plasma Mobile: next-gen Linux phone
036 The future is open source

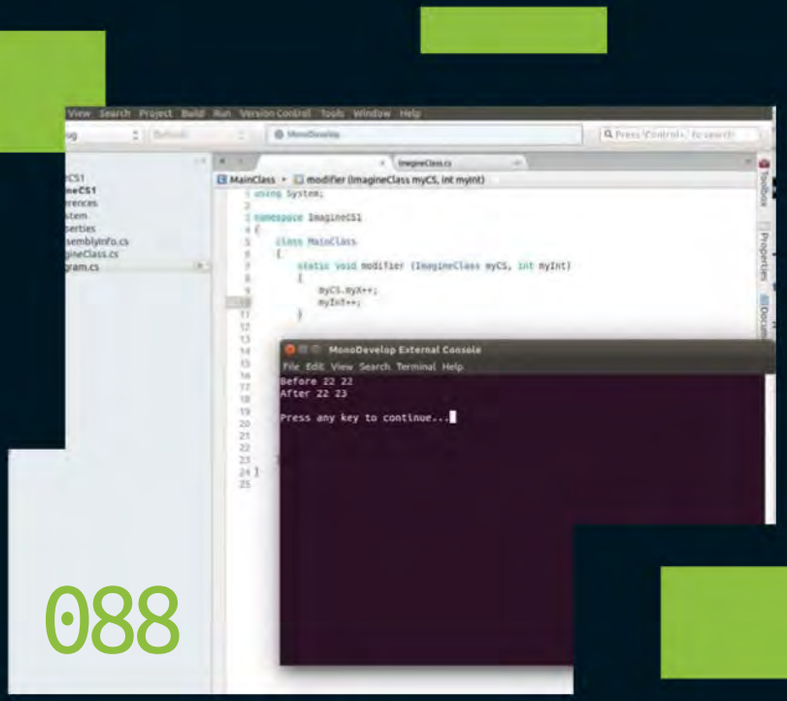


Enhance your system

044 Build your own system tools
052 Troubleshoot & Repair Linux networks
058 Turn an old PC into a NAS box
062 Debug your own Linux software like a pro
066 Automate your audio in Linux
070 Set up game controllers to work on Linux
074 Speed up your system by combining SSD and HDD

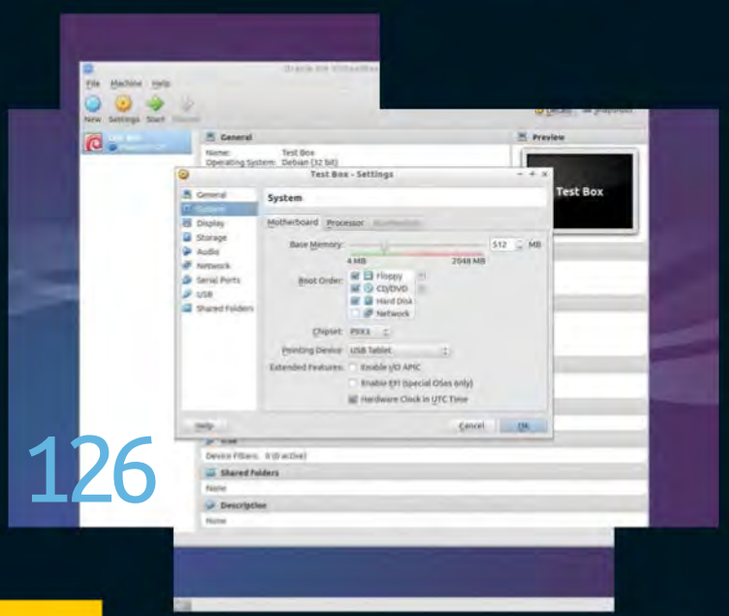
Programming in Linux

080 100 ways to master the command line
088 Start programming in C#
092 Use your Kinect with Linux
096 Intermediate AWK programming
100 Launch your first Docker container
104 Deploy software to Docker containers using Deis
108 Run science experiments on the ExpEYES kit
112 Program a Parrot AR.Drone's flight path
116 Home automation with your Pi



Harness FOSS

126 How to virtualise Linux
134 How open source giants are changing the world
138 Running Windows apps with Wine
144 20 LibreOffice tips & tricks
148 Bodhi Linux 3.1.0
150 Korora 22
152 OSMC
154 Mageia 5
156 Solus Beta 2
158 KDE Plasma 5
168 Code Gnome software




PL
KeePassX
sqrt[x]
PL

BEST FREE SOFTWARE

ALL THE SOFTWARE YOU NEED TO USE AT HOME & WORK

Using Linux for any kind of computing purpose is genuinely feasible these days, unlike the dark ages of the Noughties. With excellent, large communities around different distros and software, many apps and programs have been created that can fit a wide variety of purposes.

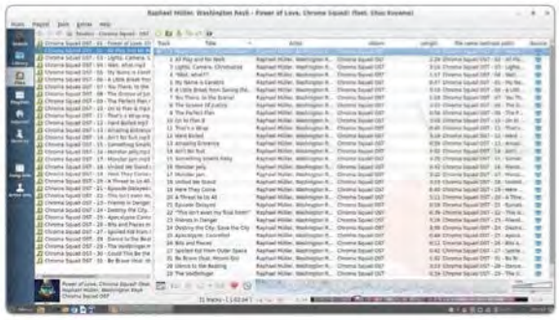
While some of the best apps may rise to the top of the pile, you may not always be looking at that specific pile. With that in mind, we've put together a guide for the very best Linux apps that you can get for all of the major uses. From office workers to gamers and even scientists, there are plenty of great pieces of software for anyone wanting to use Linux.

MULTIMEDIA & ART

Play games, paint, make music – the choice is yours and you have plenty of it

Even while working, you may need a way to keep yourself focused – listen to some music, or take a quick break and watch something. Some people's work might be more media-centric. For whatever reason, Linux has a wide range of free media apps to help you relax or be creative

CLEMENTINE MUSIC



Our favourite music player and manager is also definitely one of the best around. Clementine, as well as being able to simply play music, enables you to easily manage your entire media library, playlists and even some content from your online music streaming services. It has more features than any other music player and it is the only one we like on Linux that can create smart playlists for you. And, of course, it plays every type of music file you can think of.

AUDACITY AUDIO EDITING

Audio editing may be a bit niche, but Audacity is one of the best tools for the job. With powerful effects, track control and a great workflow, it is easy and quick to edit or produce whatever you're working on, whether it's music, podcasts or sound effects. Our top tip is to make sure you know your way around your sound server on your system for different microphone configurations.

BLENDER 3D MODELLING

A long-running and great piece of software, Blender is your one-stop shop on Linux for creating 3D models and 3D animated films. The quality of the Blender-created shorts over the past few years has been incredible, demonstrating that you can do just about anything with it if you try hard enough. The dev community includes a lot of people that use it professionally, so it's in very good hands and has been made with 3D-modellers in mind.

KRITA DIGITAL PAINTING

While GIMP is excellent as an image editor, and you can definitely use it for digital colouring, Krita is where you can do some really beautiful art. The layout and workflow is better optimised for painting than GIMP and also has better tablet support. Instead of doing flats, you can actually paint with a full colour palette that is easier to select from. In addition, there is a better selection of brush types and effects.

VLC VIDEO

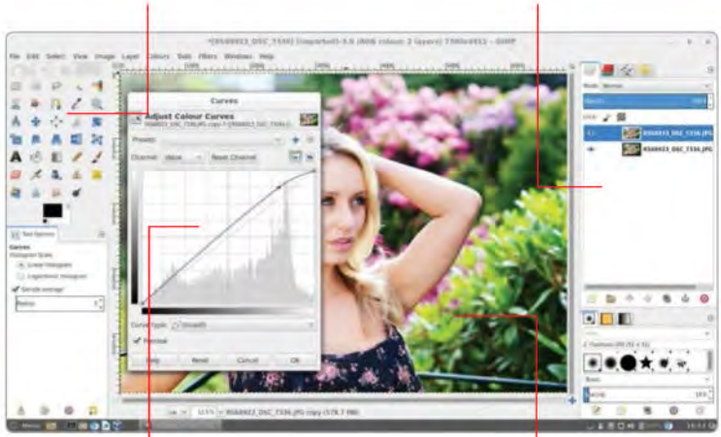
VLC is ridiculously good. Video playback on any operating system has always had its ups and downs, requiring you to scour the Internet to find a way to play some obscure video file. On Linux, this sometimes required switching between media decoders, but VLC does away with all of that by having all the codecs built in. And we do mean all of the codecs – it will play anything you can throw at it. That's before you get to its full network streaming capability.

GIMP IMAGE EDITING

Photoshop shmotoshop. GIMP may not cost you a kidney, but it can certainly do just about anything Photoshop can. It isn't exactly fair to refer to it as a free Photoshop because it is an image editor in its own right, with its own individual workflow and interface. It has a great dev team behind it and an active community, so it is easy to learn how to use GIMP or move over from Adobe's offering.

Here we have the main manipulation tools – your standard paintbrushes and drawing tools, along with a variety of selection tools along with size, perspective and placement tools

Here is your work space – you can use the layer view for an overall look at the composition, but you can also look at the alpha channels, recent documents and action history



Many effects, such as the colour and filter effects, are kept in separate windows so you can modify the properties and get a small preview of how it will change the image

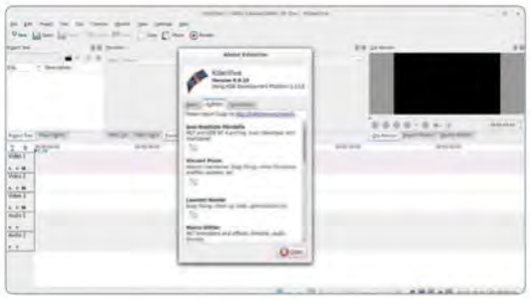
Use all these tools to make photos just that little bit better, with touch-ups, white balancing and maybe even an adjustment layer or two to make them really pop in colour

SHOTWELL
PHOTO MANAGER

GIMP can be used to edit photos, but what can you use to organise them? This is where Shotwell comes in, a photo management app that can help you organise photos into albums on your system for easier navigation, upload them to certain social media sites to make the post process easier, and it can also perform light batch operations on photos with basic editing techniques. If you like to take a lot of photos, this is probably for you.

UFRAW
RAW EDITORS

When shooting photos with a really good camera, you'll usually be able to get RAW photo files out of the SD card. These aren't compatible with GIMP and some other image viewing software as they first need to be processed. UFRaw is a free piece of Linux software that lets you load a lot of the major RAW files and start playing around with the white balance and other aspects of the photo, so you can create a JPEG of the perfect version.



KDENLIVE VIDEO EDITING

Editing videos in Linux is generally better for normal folk than it is on other operating systems, as the level of free software on Linux is higher than the likes of iMovie and Movie Maker. KdenLive is a prosumer-grade video editor with enough functions to be almost on par with Adobe Premiere. Not only will it let you arrange videos in a linear timeline, but it also has a wide array of effects, including green screen effects or chroma key.

INKSCAPE VECTOR GRAPHICS

Making vector graphics is very different to digitally painting an image. Vector graphics are used to adjust art to any size without weird pixelation. Inkscape is the most popular one available on Linux as it has a powerful set of tools to help you create vectored artwork for any application, including Bezier curvers, node editing, boolean operations on paths and many more.

EREADERS

Reading books on your computer can be very convenient and a good ebook manager makes the task much more hassle-free and relaxing. Calibre is definitely the best for the job on Linux, as it is compatible with every major ereader format, such as EPUB. You can also use it to organise and read normal document formats as well. It's optimised to run on small screens if you want it to, so if you fancy creating a custom mini Linux ereader, Calibre is the software you'll need!

GAMING ON LINUX

Not sure it's a thing? Think again – here are some awesome free games

DOTA 2

The sort-of sequel to a free mod for a game made by a different company, *Dota 2* has a weird origin. Once you get past that though, you find an excellent MOBA that makes it a great alternative to *League of Legends*. It has been fully embraced by the eSports community too, with regular high-stakes tournaments and leagues to play in. If you're not the highly competitive type though, you can still enjoy it casually.

MARVEL HEROES

We swear to you that *Marvel Heroes* is actually pretty good, despite what your initial assumptions might be. The game lets you be one of an ever-expanding line-up of Marvel characters and heroes as you repeatedly left-click on bad guys to try and save the world. It takes its cues from the comic books and the current movies, so you can use Mark Ruffalo's Hulk if you really want to. Don't get too hooked into its freemium trappings though.

TEAM FORTRESS 2

Part first-person shooter, part crazy economical experiment, *Team Fortress 2* was Valve's first free-to-play game and they made a ridiculous amount of money from it – and still do make it, thanks to the hats you can find in-game. You don't need to pay a penny to have a good time though, as it has a variety of classes and game modes that suit any player and excellent core gameplay mechanics to carry it while you shoot snipers or robots.

STAR TREK ONLINE

We will spare you the *Star Trek* references to tell you that *Star Trek Online* is a great MMO that lets you fly around in a starship shooting phasers, or beam down to other planets so you can... shoot more phasers. The core game is free these days, but there are paid expansions for extra content that expands the *Star Trek* universe with new stories. Also, you can set your warp speed to two decimal places, which is completely pointless but also quaintly excellent.

STEAM GAMER? THESE ARE ALL THE LINUX-COMPATIBLE GAMES:
bit.ly/1WjoYaJ

OFFICE SOFTWARE

Set up the ultimate office system with these powerful apps

LIBREOFFICE

The successor to OpenOffice has long since established itself as the best Linux office software around, with just about every feature you'd expect from something like Microsoft's pervasive product. Here's a selection of the best applications included in LibreOffice and what they can do

WRITER WORD PROCESSOR

There's a standard word processor in any office suite, however Writer does a lot more than other text editors you can find in Linux. It has all the formatting tools you'd expect, such as layout functions, advanced macro and mail merge tools that you find in the professional software versions. It does prefer to save files as .ODT, but you can change it by default to .DOC or .DOCX to be compatible with Microsoft.

CALC SPREADSHEETS

The equivalent of Microsoft Office Excel, Calc has all the advanced features of Excel without you needing to relearn how to create different formulas and codes. The workflow is similar and you can edit the formatting for individual cells. You can even do Pivot Tables for large data operations. Compatibility with Excel files is okay, but not great.

IMPRESS PRESENTATIONS

LibreOffice's PowerPoint may not be the flagship app of the suite, but it's able to create and replicate the kind of presentation style you're used to, as well as being able to open and display pre-made presentations from other software. It has limitations in the way it uses outside media, relying on the codecs and streamers available to it, and as a result, this can change as a file is moved between systems.

MATH FORMULAS

Similar to the way TeX and LaTeX enable you to write and draw mathematical formulas, the Math app allows you to create equations to input into any of the main three pieces of LibreOffice software. As Writer has a free PDF converter, you can even use it instead of a LaTeX editor in a few situations. It's quite a minimal program though, focusing on easily writing formulas in a graphical manner.

WHY NOT OPENOFFICE?

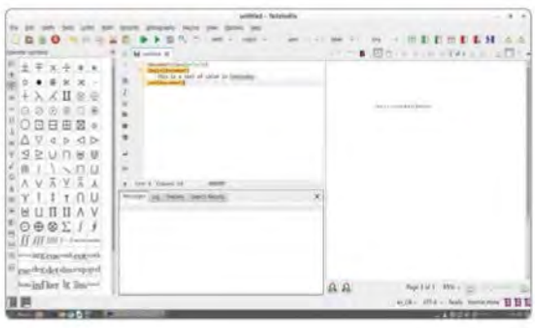
Here, we're concentrating on and recommending LibreOffice as it really is the best office suite on Linux. You may be wondering why we haven't mentioned OpenOffice; well, OpenOffice underwent a lot of changes to the dev team a few years back, with most of them leaving to create LibreOffice from the current state of OpenOffice. Development has therefore stagnated somewhat since then, while LibreOffice has gone through many updates and overhauls to keep it relevant. Because of this, it's now the better of the two and definitely much better than KDE's Calligra Suite.

TEX & LATEX

TeX and LaTeX are programming languages used to create documents – think of it like a very manual word processor. The benefit of this is that you can control exactly how a document will look and they can be easily written as PDFs. Moreover, you can customise config files and scripts so that you can do some initial setup and then rely on your created classes to quickly style up text.

01 Consider TeX Studio

With better placement of images, mathematical formulas and more, TeX and LaTeX are great for academics and professionals. We quite like to use TeXStudio to create documents –it's an "integrated writing environment" that brings things like syntax highlighting , multiple cursors, bookmarking, image drag-and-drop and more.



02 Write your document

Writing documents in TeX requires you to understand the syntax of the code, but it's a very powerful thing once you are familiar with the documentation. You can bring up shortcuts to create mathematical formulas, and you can keep a running preview of what you have created so that you can go back and check for any errors.

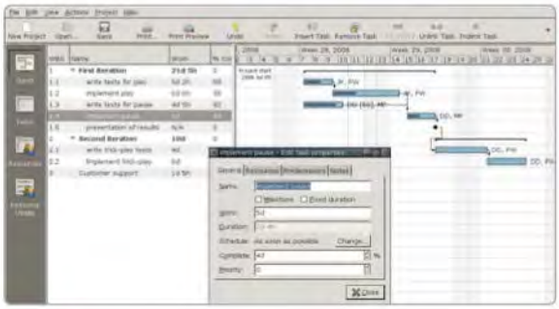
03 Save your document

As you're writing your document, you can save it as a .TEX file to go back and modify it, however once you're finished you can output it as a PDF for presentation. Not many word processors allows you to do this properly, but TeX makes it easier and better-looking in PDF form.

PROJECT MANAGEMENT APPS

Help your work run smoother with these Linux project managers

GNOME PLANNER



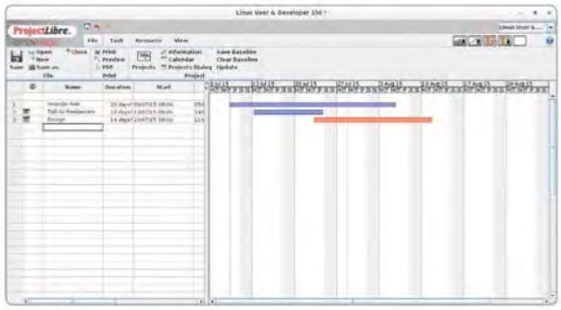
A GNOME app that integrates well with other GNOME and GTK software, Planner is a simple project manager that enables you to easily create plans and graphs for an entire project. It lets you create Gantt charts, manage resources and more with its plain interface, a common feature of current GNOME apps. All its data is stored in either XML files or a PostgreSQL database, and you can even export the whole thing on HTML to move it somewhere else. The software is also cross-platform compatible, in case you need to edit it away from a Linux machine. It's well liked in the community and it's also one of our favourites.

-VS-

THE VERDICT

ProjectLibre is the much more powerful tool of the two softwares we're comparing here, giving you much more control over how you create a workflow and how to qualify it. Planner is still very good, but it may be more useful for people doing personal projects than those looking for an office tool.

PROJECTLIBRE

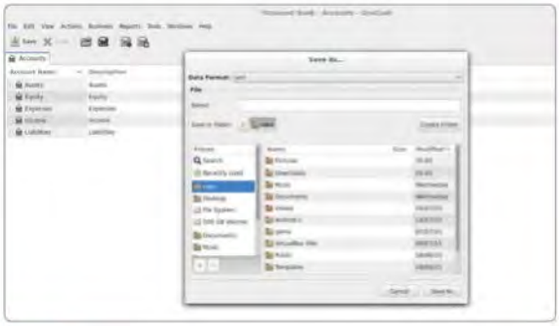


Positioning itself as a full Microsoft Project replacement app, ProjectLibre is a bit more complex than Planner. As well as all of Planner's functions, you can use both PERT or RBS charts, and perform both cost-benefit analysis and earned value costing to really get a hold of a project and its resources. You won't find ProjectLibre in every Linux repository, so you will have to look for it online on SourceForge or via its official website, but it's easy enough to install with binaries available and the source code if you fancy building from scratch.

HOME & OFFICE ACCOUNTING

Manage your finances at either of your desks with these top finance apps

GNUCASH



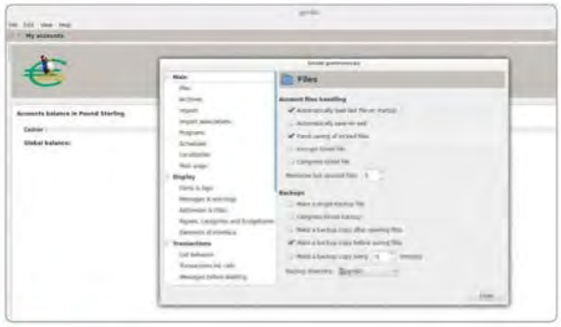
Easy to use, powerful and flexible to your own needs, GnuCash is great for personal finances and business finances. As well as tracking your bank accounts, income and expenses to help properly organise your cash and savings, you can also keep an eye on stocks so you have a better idea of your current assets. All cash flow requires double-entry; debit and credit. There's also a chequebook-style register along with a multitude of report types that you can generate in case you need to visualise the accounts or submit a report.

-VS-

THE VERDICT

Because of all it's extra features, including the stock options, GnuCash is definitely the better choice for those who really need a powerful accounting software to look after their business finances and more. Grisbi is still good, but GnuCash is still going very strong after many years.

GRISBI



Grisbi is a slightly different financial manager to GnuCash, angled a little more at home users rather than big business. It's cross-platform like GnuCash and lets you properly track transactions between your accounts. It doesn't have any stock information, but it enables you to view and create reports, as well as set up budgets to see how close you are to keeping with them. The reports function allows it to be used in a more small-business scenario and its slight simplicity over GnuCash might be useful to some.

WEB & PRIVACY

Get the best software available for connecting to, downloading from and for talking over the Internet

HEAVYWEIGHT BROWSERS

Most of you are using one of these – but which is really the best?

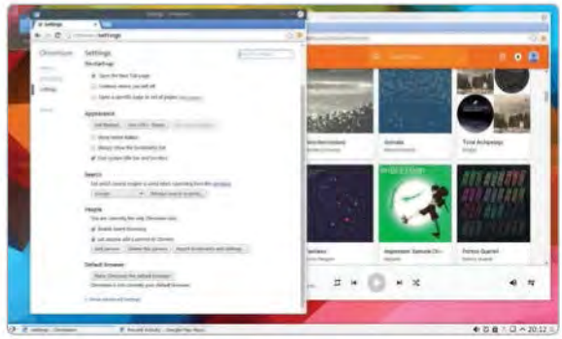
FIREFOX -VS- CHROMIUM

Firefox is pretty much ubiquitous to Linux, if not in its vanilla form then as a distro-rebranded spin like Debian's Iceweasel and GNU's IceCat, and is most famous for its incredible range of extensions. It also holds to a pretty fearsome pace in terms of its update schedule, although these updates often introduce new user-visible features as well as backend tweaks and fixes. May, for example, saw the introduction of tab-based preferences; June saw the integration of Pocket and a new Reader View mode; Firefox Share was integrated with Firefox Hello in July, enabling users to invite people to the Hello VoIP service through their social networks. Firefox also provides Extended Support Releases that are kept free of large, disruptive feature introductions and only receive major stability and security updates.

THE VERDICT

	Firefox	Chromium
FEATURES	9	8
MEMORY EFFICIENCY	8	6
EXTENSIBILITY	10	8
OVERALL	9	8

Firefox's extensibility is a massive plus for the open source browser and it's better memory-wise than Chromium. Both these make it just that bit better in our estimations.



The open source bedrock of the increasingly popular Google Chrome browser, Chromium is actually incredibly similar to Chrome, but with a few notable differences. Essentially, Chrome is 99% Chromium with the addition of some proprietary elements, such as Flash (although its days are numbered). Chromium uses the open source media codecs – Vorbis, Opus, VP8 and VP9, Theora – and then Chrome adds the proprietary MP3, MP4, AAC and H.264 on top. Another important difference from Chrome is that while Google pushes updates out to the user automatically, Chromium relies on the user or maintainer to keep Chromium fresh. There can be some variation with Chromium, with some vendors adding those proprietary codecs themselves, but you can always find the latest pure build at **download-chromium.appspot.com**.

PIDGIN

Facebook and Google recently ended their support for the XMPP API, which means that messaging clients such as Pidgin can no longer officially connect to the services. A new Google solution is in the works but there's no word from Facebook yet. However, James Gebloski provides a work-around plugin for Debian and Ubuntu that will help you connect the Pidgin client to your Facebook account.

01 Plugin setup

First navigate over to /etc/apt/sources.list.d and then create the file 'jgebloski.list'. Open it up in a text editor and add the following line:

```
deb http://download.opensuse.org/repositories/home:/jgeboski/<version> ./
```

... replacing <version> with one of the following, depending on your distro: **Debian_8.0**, **Debian_7.0**, **xUbuntu_12.04**, **xUbuntu_14.04**, **xUbuntu_14.10** or the newer **xUbuntu_15.04**.

02 Repo key

To add the repository key, enter the following commands into the terminal:

```
wget http://download.opensuse.org/repositories/home:/jgeboski/<version>/Release.key
sudo apt-key add Release.key
sudo rm. Release.key
```

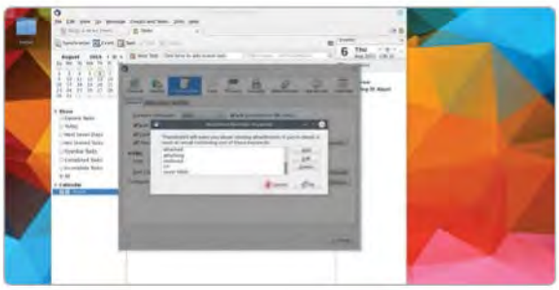
03 Non-XMPP account

Now you just need to run a **sudo apt-get update** and then a **sudo apt-get install purple-facebook**. Next, restart Pidgin. Add a new account or modify your existing one, pick 'Facebook' for the protocol – not 'Facebook (XMPP)' – enter your username and password. Leave Local Alias blank.

DESKTOP EMAIL CLIENTS

Web clients are hugely popular, but desktop clients are very powerful

THUNDERBIRD



Despite concerns in recent months that the project is dead or dying, Thunderbird is very much alive and kicking. The project didn't see any major feature introductions for a while, although this is chiefly down to the development work moving across to the community following the Mozilla chair's announcement that the company itself would no longer be developing Thunderbird – it has effectively gone the way of SeaMonkey. But it is still one of the best and the most widely available desktop email clients, and new features such as the Lightning Calendar add-on and OAuth2 support in Gmail were added recently.

KMAIL

If you use the KDE desktop, the most powerful, most configurable email client available to you is already installed: KMail, which is heavily integrated with the Kontact application. KMail provides every standard feature you can think of and then goes on to provide advanced features that blow the competition out of the water – everything you can do with a plugin in Thunderbird is an option in KMail's preferences. It has incredible search and filtering tools, great conversation threading, robust and in-depth security settings, integration with other KDE apps like KWallet and external spam checkers like SpamAssassin, and the entire software is also completely configurable to your own taste.

PRIVACY AND SECURITY TOOLS

Protecting your online activity and your personal data is becoming more important than ever

KEEPASSX

In the interests of security, we always recommend using different alphanumeric passwords for each of your online accounts (although having a base, perhaps phrase-based, password and then creating memorable permutations for your various accounts is another good move). Keeping track of them all can be a pain, so for simplicity and security you can use KeePassX. Store all your sensitive data inside an encrypted database, and keep it inside Dropbox or a secure server. You can then access that database from other devices and use a single master password to unlock everything you need from whichever device you're using.

TAILS

We mention Tails a lot when it comes to security and privacy software, but with very good reason. It is without question the best distro out there for giving you a fully-protected online experience that's ready to go out of the box, with the Tor network, Tor Browser and I2C set up for immediate use. The system lives entirely in your RAM and wipes itself after use, leaving no trace on the hard drive you're live-booting on top of, and isolates applications with AppArmour, can spoof your MAC address and automatically encrypts your communications.

PGP MAIL
Send secure, encrypted and signed emails without having to install any extra software, and keep your communications private from anyone that may want to snoop on what you're saying

CAMOUFLAGE MODE
Using Tails out and about, and don't want to draw attention? You can launch it in a Windows camouflage mode that makes it look exactly like Microsoft's OS, but still has the same functions

DAY-TO-DAY SOFTWARE
Even though it's hyper secure, you don't have to sacrifice usability. Tails comes with all the best software you could need

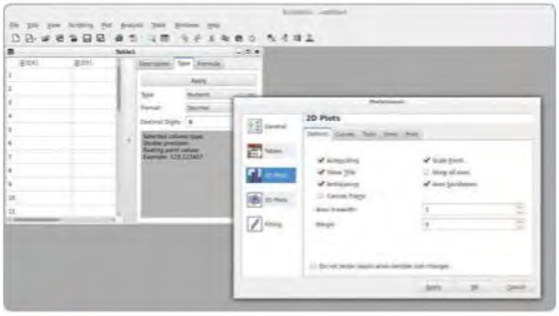
DON'T LEAVE A TRACE
When you shut down after a session of Tails, the RAM will be completely erased so that no one can try any advanced forensics on it

TOR NETWORK
Tails automatically connects to the private Tor network, and you can even see the map of how you're connected and reset the connection if you want to

SCIENCE SOFTWARE

Linux has incredible FOSS for doing proper science and engineering work

SCIDAVIS



Software for scientific data analysis and visualisation (the name is a weird acronym), SciDAVis enables you to manually input or import data from a variety of sources in order to be analysed via the various statistical methods built into the software, then plot 2D and 3D graphs, matrices and more that are suitable for publication. It's also cross-platform, so if you need to work on a variety of different machines, everything will work between them.

STELLARIUM

Going out stargazing is a lot of fun (just ask *All About Space*), and while you can see some great things by pointing your telescope at a random spot, to make the most of it you need to plan – Stellarium is perfect for that. All you need to do is set your position, give it a time and you will be given an annotated view of what will be in the night's sky. Note down the co-ordinates (if your telescope is fancy enough) and you're sure to have a great night stargazing.

CAIN

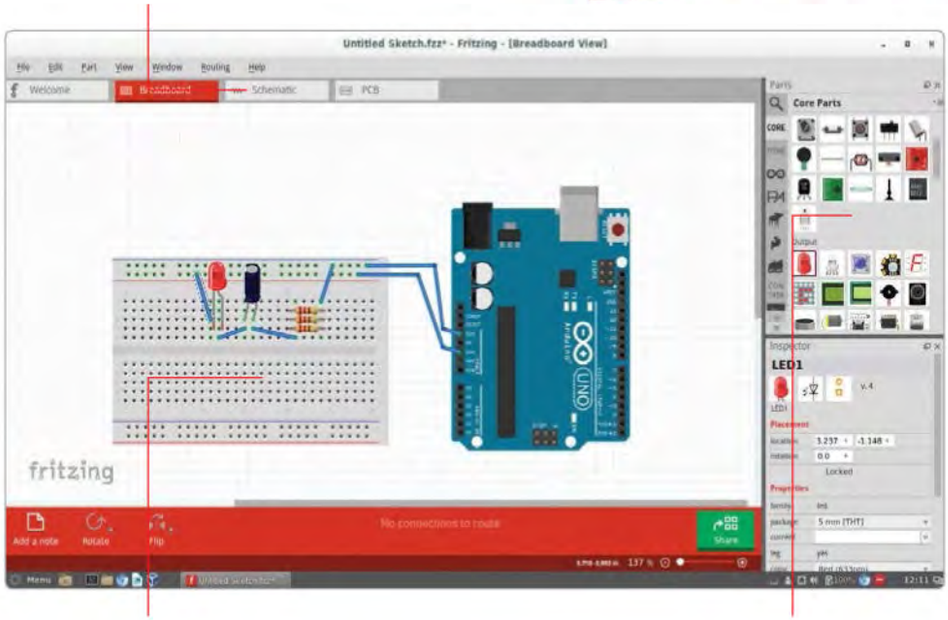
This is a piece of software useful for performing stochastic and deterministic simulations of chemical reactions. If that at all sounds interesting to you, then you may like to know it can also solve models using Gillespi's direct and first reaction method, Gibson and Bruck's next reaction method, Tau-leaping and a few more. It can import and export relevant data to make analysis easier for you, using XML and SBML formats for this.

PLANETS

If you've ever tried to create an orbital model of a series of planets, moons and one star (such as in our own solar system), you'll know that coding the mechanics can be a right pain. Luckily, there are plenty of programs out there like NASA's GMAT that can help you model orbits without the need for doing it yourself. This can be useful for teaching yourself or others about how celestial bodies move.

NEW TO ALL THIS? START WITH THE MATHEMATICA: bit.ly/1jTTbds

View the circuit you're building as a schematic, or build one directly as a schematic to begin with. You can even create a PCB view



You can arrange your components and microcontrollers as you would in a real circuit to get an idea of how you should make it

Choose from hundreds of different components, microcontrollers and more. Change their settings, rotate them and customise them to your preferences

FRITZING

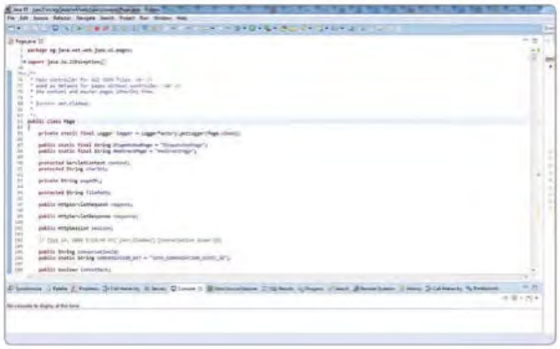
A great piece of software for planning out or sharing electronic circuits, Fritzing lets you create custom circuit paths not only using standard components such as LEDs and resistors, but also with a selection of different microcontrollers from across Arduino's range, as well as the Raspberry Pi. It also has a neat trick of turning the planned-out images into standard electronic component symbols to make sharing the exact layout of the circuit much easier. Even further, you can use it to design PCBs to then be printed and used by yourself or others. It's really the best tool for anyone doing electronic design to use as it makes the whole process that much easier. If you want to get into circuit design, it also has an in-depth example and tutorial section available to teach you how to use it.

PROGRAMMING TOOLS

Make apps and solve the mysteries of the universe with these amazing tools

TOP IDEs

Coding is a mainstay of Linux activity and no matter which language you're writing in, you're going to need an integrated development environment that's stocked with all of the right tools and features. You'll need plugins, intelligent formatting, debugging tools and more. Basically, you need these IDEs...



ECLIPSE C/C++

A very popular and powerful IDE, Eclipse is perfect for coding in C and its derivatives on Linux – frankly on any other operating system too. As well as being cross-platform, it has a deep and varied plugin system that will enable you to customise the way Eclipse works. It also adds extra languages in case you really like the layout and want to try other tasks too. What is more, it has powerful debugging and compiling tools as well.

IDLE PYTHON

While this is the standard development environment for Python, it's also very good at it. Our favourite feature is the shell: a working python environment where you can try out bits of code to work out what does and does not work, along with running the entire code without the need for compiling. It hooks in well to any custom modules you might have made for a project, giving you the arguments for functions within.

INSTALLING ECLIPSE PLUGINS

Eclipse has fantastic support for plugins, and a rich library of plugins you can access or download and install manually. Go to Help> Install New Software and then Add. From here you can add a repository for the Eclipse plugin you want to install; this will allow the plugin to stay up to date. Alternatively, you can use the Add function to install a plugin directly from a ZIP file. Be aware though, you won't get the same updates this way. You can head to the Eclipse marketplace (**marketplace.eclipse.org**) to find a great selection of all the plugins you can use.

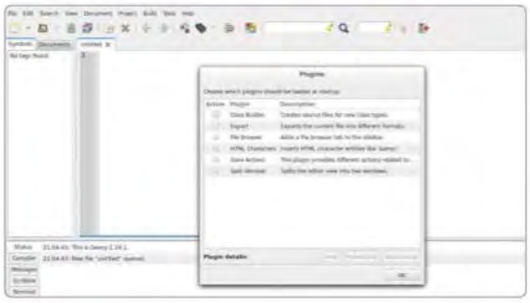
RADRAILS

RUBY/RUBY ON RAILS

A standalone IDE or plugin to Eclipse, RadRails is a flexible IDE for working on Ruby projects. It's great on its own if you work exclusively in Ruby, with an integrated debugger and other helpful features, such as code assist and structure hierarchy to make navigation easier. The Eclipse plugin concept is great, as it enables you to use Ruby alongside other code in a familiar environment.

INTELLIJ JAVA/JAVASCRIPT

Part of a network of IDEs, IntelliJ prides itself as being the most intelligent Java IDE (whatever that means), and in our experience it is pretty great. As well as standard smart code completion that gives you suggestions and lets you know what arguments a function needs, it checks code quality and senses any problems. It's good for web-based Java and creating mobile applications.



GEANY WEB

Developing for the web is different to developing normal programs. For starters, it's not as easy to test changes locally. There are also a variety of ways you might make a website and a selection of different programming languages. However, many IDEs can help you code in these various web languages, along with code mark-up and hierarchy interfaces to help navigate easily. Eclipse is a good bet for this thanks to its plugin nature, but you can also try a light IDE like Geany that should do it almost as well.

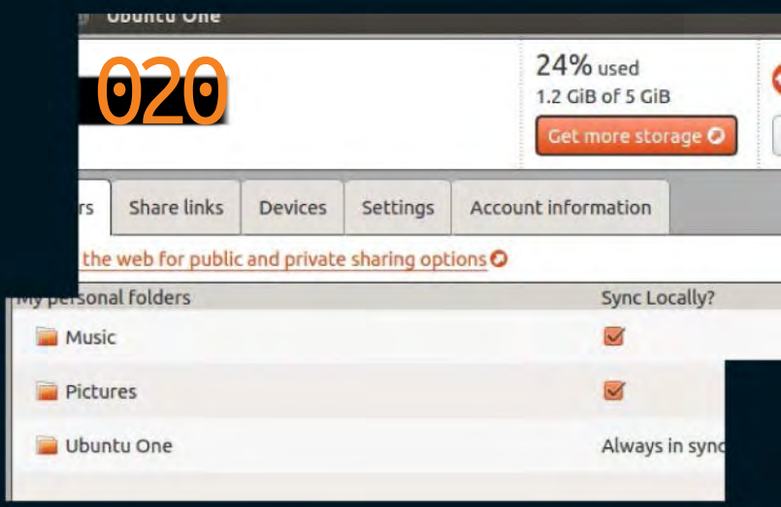
Open source world

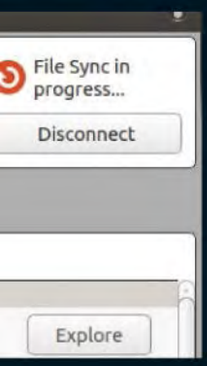
020 Evolution of Ubuntu

028 Get started with Ubuntu Phone

032 Plasma Mobile: next-gen Linux phone

036 The future is open source





036

028

"LinuxCon Europe 2015 saw plenty of announcements of projects and more"

The Evolution of Ubuntu

We celebrate over a decade of the distro by looking back at Ubuntu with its movers and shakers

Looking back 11 years, a seemingly unassuming distro nicknamed Warty Warthog emerged in the Linux landscape and set in motion a cultural landslide that would see Linux rise from the shadows of the archetypal operating systems and become a household name. Since October 2004, Warty Warthog has evolved through numerous forms into Wily Werewolf, the latest version of Ubuntu.

A catalyst for change, Ubuntu has achieved a great deal in its first fabulous decade. It pioneered the idea of a Linux operating system that just worked straight out of the box, without the need to manually troubleshoot and configure your hardware. It popularised the graphical interface that we use for most of our distro installations today, making them more accessible and easier to use, as well as the long-term support releases that many of us rely on for our main computers. It sharply divided opinion by having an opinion on the future of desktop and mobile operating systems, predicting convergence and boldly taking a hand in preparing the way.

Two forces are driving Ubuntu forward that, together, have found a degree of success in getting Linux into the public eye that very few other distributions have achieved: Canonical's leadership and the Ubuntu community.

The relationship between the two is an exemplary model of how open source companies can collaborate with their end-users in order to reliably deliver exactly what they want. Ubuntu contributors knows that they are heard by the Canonical team leaders, and the development process is incredibly democratic while still being rigid enough to adhere to Ubuntu's strict release schedule and LTS commitments. It's a delicate balance and Canonical struck it cleanly, producing quality Linux distros twice a year.

Here we look back at 11 years of Ubuntu and celebrate the achievements of the developers and contributors who made it real for us – and look to the future, to see where we're going.

»

Jane Silber
Chief executive officer
Jane joined Canonical before the original Ubuntu release as the chief operating officer. She's worked on many Canonical projects and has a C and C++ programming background.

Martin Pitt
Software engineer
Martin was a prolific Linux developer for seven years before becoming joining the first Ubuntu developers. He now works in the plumbing layer, but used to do more for the distro.

Alan Pope
Applications project manager
A veteran of the Linux community, Alan was once a tech support volunteer on Launchpad. Now he coordinates with the community in making core apps for Ubuntu phones.

Richard Collins
Mobile product manager
Richard heads up the mobile side of Ubuntu. Before coming to Canonical he already had a mobile background from working at the Symbian foundation for many years.

Ubuntu the first

Warty Warthog was released on 20 October 2004 to a much different Linux world than today

There wasn't much branding on the original Ubuntu desktops. You can see here that it uses GNOME 2 as its preferred desktop environment

The original Ubuntu comes with a selection of default apps that look like a slice of open source history. GIMP is just one example

Ubuntu was one of the first distros to automount flash drives but you could still do it the – now literally – old-fashioned way

Like today, Warty Warthog shipped with Mozilla Firefox, although it was in one of the earliest forms of the browser

It's February 2004 and it's a very different world from what we know today. Windows XP is the latest desktop from Microsoft. Apple hardware is still running on PowerPC. BlackBerrys and PDAs dominate the smartphone market and x64 chips are just about making their way to market. Linux is barely in the zeitgeist outside of developer circles, but this is soon about to change.

"I was one of the first people Mark Shuttleworth called back then," said Martin Pitt, software engineer, who was there at the start of what would be a major revolution for the Linux desktop. "He sent me an email explaining his idea about doing a Debian-based Linux project with a regular release cycle that was also user-friendly."

This was the beginning of the development of Ubuntu 4.10, code-named Warty Warthog, and of Canonical itself. At the time, Linux desktops were very different. There were no graphical installers, there was no proliferation of user-friendly distros and very little push for 'normal' people to start using Linux.

"Canonical and Ubuntu officially started in April of 2004," elaborates Jane Silber, Canonical CEO. "[Mark] pulled a group of about ten people together with this vision of creating Ubuntu... I met him a couple months later in July and just immediately believed in the vision of Ubuntu and Canonical."

Jane later joined as the COO of the company and, after several months of development, the very first Ubuntu was released on 20 October 2004. The release was a bit slow catching on though, explains Martin:

Ubuntu timeline

20 October 2004

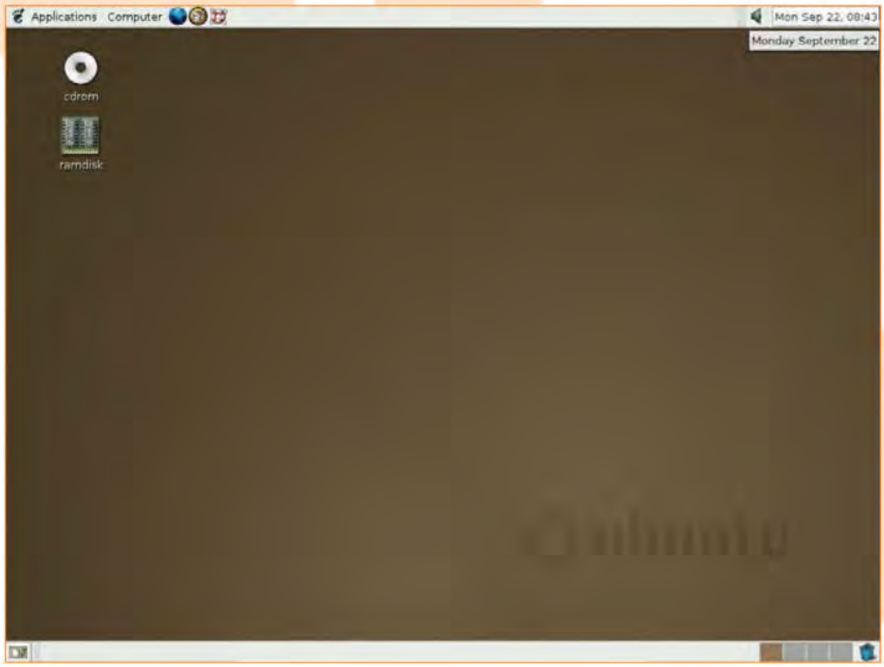
In the beginning...

Ubuntu 4.10 Warty Warthog is released. Based on Debian testing and using the GNOME desktop environment, it was a solid first release. It wasn't an overnight sensation but it found a core group of loyal users.

18 April 2005

The first spin

Kubuntu was the first alternate flavour of Ubuntu, releasing shortly after Ubuntu came out and offering the KDE desktop environment as opposed to Ubuntu's default GNOME.



Left The basic GNOME desktop from the time. The live disc and install disc were once two separate images

The Linux landscape

Alan gave us an apt description of how Linux was in 2004

"At the time I had a Phillips webcam; on both Red Hat and Debian there was some trouble with it and I used to have to keep recompiling my kernel. That seemed like the thing that people did in those days. Something doesn't work, so you then get the kernel source and you run these obscure commands that may or may not work and may take a long time; eventually you get kernels that may or may not work. It may not even boot and, if it does boot, well maybe your webcam works, maybe it doesn't. I installed Ubuntu and my webcam just worked out of the box. I didn't have to do anything and I thought this [has] got to be the way forward."

"Development up until the first beta had been somewhat secret"

"To be honest it was still pretty much by developers for developers, so the immediate coverage was quite low. I prodded some famous German computer news magazines and [at] first they said, 'Yeah it's just another distro – why should we report about this?'"

Development up until the first beta had been somewhat secret and while the first release may have 'only' scored a few thousand users, its name started to grow throughout the Linux and open source community.

"I was in my local Linux User group in Hampshire and one of my friends knew that I was quite into Debian," Alan Pope, applications project manager, recalls. "He mentioned this thing, this new thing that was being developed by some crazy South African. I'd never heard of it as I wasn't on any Debian mailing list and I wasn't really involved with the Debian community or anything. But it looked quite nice and I installed it. That was late 2004 and that's pretty much it – I've used it solidly ever since."

While a small release at the time, 4.10's legacy is massive. It was relatively easy to install, had a desktop from the start that was preconfigured and would automatically mount flash storage. These elements have since been expanded upon greatly.

The developers were proud of their work and it wasn't long before 5.04 was in development. The next two years were very important for Ubuntu and would establish it as a household name among users and developers alike.

Above The original splash with the original logo, Ubuntu's preferences for orange has not changed much over the years

8 July 2005

Ubuntu Foundation

Created by founder Mark Shuttleworth, the Ubuntu Foundation is a trust fund to ensure the future of the distro. It's currently dormant but Canonical plan to treat it as an emergency fund if necessary.

1 June 2006

First LTS released

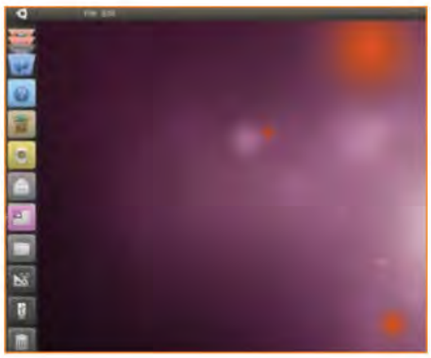
The original Long Term Support release of Ubuntu was delayed a few months from its intended April release date, but when it did come out it signalled a turning point in Ubuntu's popularity.

The early years

The meteoric rise of Ubuntu in the late Noughties caused a small revolution in the Linux desktop space

During the next couple of years a few things began to happen. Kubuntu 5.04 was released during the next cycle and is still being released today as one of the major Ubuntu spins. More features were being added by the release and, slowly but surely, Ubuntu was gaining popularity.

"It's a bit hard to say as it's always been an exponential growth," Martin tells us when we ask him which release really gave Ubuntu a foothold in the desktop Linux market. "But if I have to pick one it would be our first LTS, 6.06 Dapper Drake. That was explicitly announced for Long Term Support and we'd made particular efforts in stabilising it. It was also the first version with a graphical installer and thus made it a lot easier to do."



Above The first version of Unity was quite different to how it looks today

Version 6.06 is so far the only Ubuntu to be released outside of the months of April and October, gaining the .06 suffix after it was released on 1 June after a delay of almost two months. Support lasted for three years on desktop and five years on server, a practice that has since been stopped on LTS releases with both desktop and server versions now getting five years of support.

With the popularity came a fledgling community of users and developers. Alan has been part of this community from the start, answering people's problems on Launchpad during the free time he had with his jobs, and describes to us the evolution of the community.

"Back at the beginning, the active contributors were people who were bootstrapping things like getting IRC channels and mailing lists set up. This was all done by people with specific expertise, such as people who sat on IRC all day creating the Ubuntu IRC channels. You had people who were familiar with Mailman and they were setting up mailing lists, and people who knew about forum software would set up a forum. In the early days it was somewhat unstructured – but that was great, because we needed all of those things and it was very much a self-motivating, self-driving thing.

> "Once we got over that bump of creating the initial infrastructure, more people were able to contribute"

Above Xubuntu started to appear as one of the early lightweight spins in 2006

Someone decided 'we should have a forum' and they created one, and it became insanely popular, so once we got over that bump of creating this initial infrastructure and bits and pieces to get the project started, more people were able to contribute. It was then that contributions from the community started to take the more typical approach of doing translations, bug reporting, triage, submitting patches and producing documentation. That's the bread and butter contributions people do."

The community has continued to grow and has now become a very important part of Canonical, as Jane told us:

"The open source community and nature of Ubuntu, and cooperation between community and company, was one of the things I thought was just really unique and interesting about Canonical. I continue to think that's one of the places we've done really special work and continue to lead the field."

Ubuntu timeline

13 May 2009

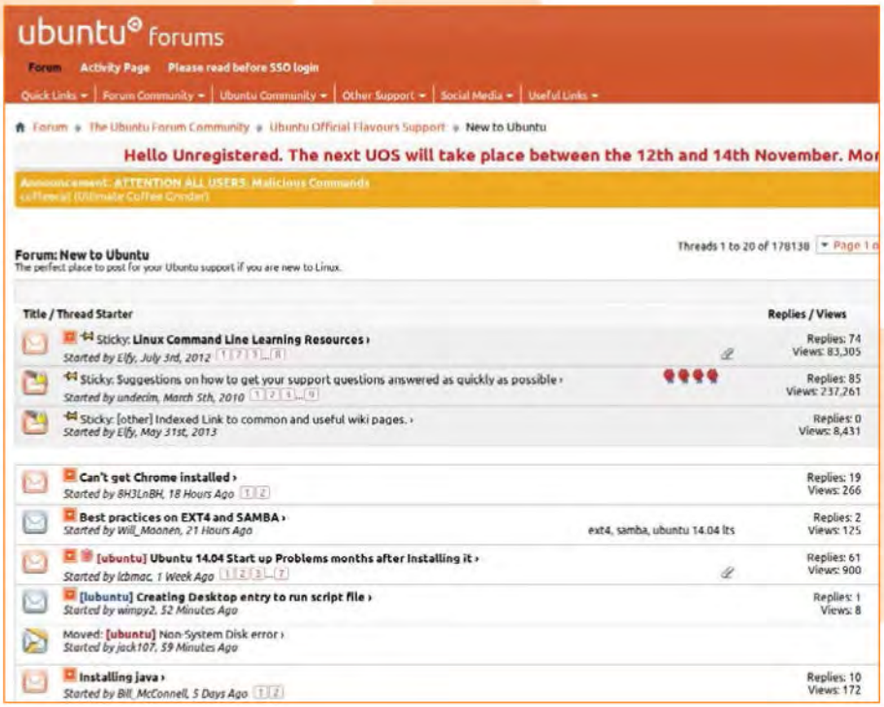
Ubuntu One launched

The cloud storage service was one of the first of its kind, originally named as it gave you access to 1GB of storage and was free for all Ubuntu users. It soon found its way into the desktop as an integrated service.

1 March 2010

Change of leadership

Mark Shuttleworth stepped down as CEO of Canonical so he could get closer to the development team at Canonical. Taking his place was Jane Silber, then the COO and still currently the CEO of Canonical.



Above The active community on the Ubuntu forums has been around for a long time

Since then, the community has worked on a number of Ubuntu projects. They've created manuals in PDF forms, translated them into different languages and started a better support site, which they named AskUbuntu (**http://askubuntu.com**).

Over the next couple of years, the community continued to grow and Ubuntu established itself as possibly one of these biggest Linux distros around, and certainly the most popular distro on desktop. In 2010, the grand vision of a unified desktop began with the release of Unity.

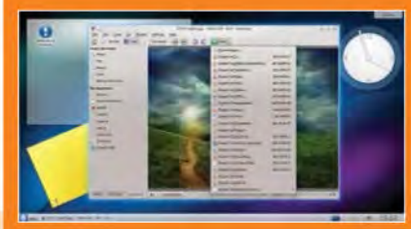
"I first saw something that looked very much like it at one of the Ubuntu Developer Summits when the design team were going over what the new shell might look like." Alan recalls. "I started using it fairly early on. It was a rocky road for about a year, year and a half. I like it, I enjoy using it."

But not everyone in the community agreed with Alan. "There were some very vocal people who decided this was not for them, they didn't like it and that's fine." Alan continued.

"I think in some ways that it has been blown out of proportion slightly and what a lot of people don't recognise is that Ubuntu ships by default on a lot of hardware around the world and Unity is the default desktop. I have no way of knowing if those people immediately go home and remove it and put KDE or Xfce on, and – to be frank – I don't really care."

Indeed, as Alan pointed out to us, there are a whole host of different desktops available in Ubuntu, in both the repositories and in the various alternate spins. Unity is at the heart of Ubuntu's design philosophy as it goes forward, though, moving towards a more unified future across all devices.

Ubuntu in all flavours



Kubuntu

The first major Ubuntu spin was Kubuntu, with the big difference being the inclusion of the KDE desktop. It was originally funded by Canonical but these days it's sponsored by a third party associated with the development of KDE. The beautiful desktop is quite different to Unity.

Lubuntu

The lighter version of Ubuntu using the LXDE desktop to maximise the amount of resources Ubuntu can use. This is useful for older or less powerful computers and can easily be scaled up to more powerful machines if you prefer the simple interface and want to get the most of your CPU.

Edubuntu

An educational spin on Ubuntu that supports the LTSP thin client so students can run the exact same distro at the same time. It's designed to be easily usable by teachers who want to create a computer lab or online classroom. It uses Unity and a variety of useful open source software.

10 October 2010

First taste of Unity

Unity was released with the netbook version of Ubuntu 10.10, giving users their first look at Canonical's new desktop environment. This was also the last release of the netbook version before it merged into the main distro.

28 April 2011

Unity for all

11.04 was the big release for Canonical as it began the company's vision of a unified desktop under the Unity environment. Based upon the then-new GNOME 3, 11.04 was in some way a GNOME shell alternative.

Now and into the future

How the future and the face of Ubuntu is being shaped right now by Unity and mobile

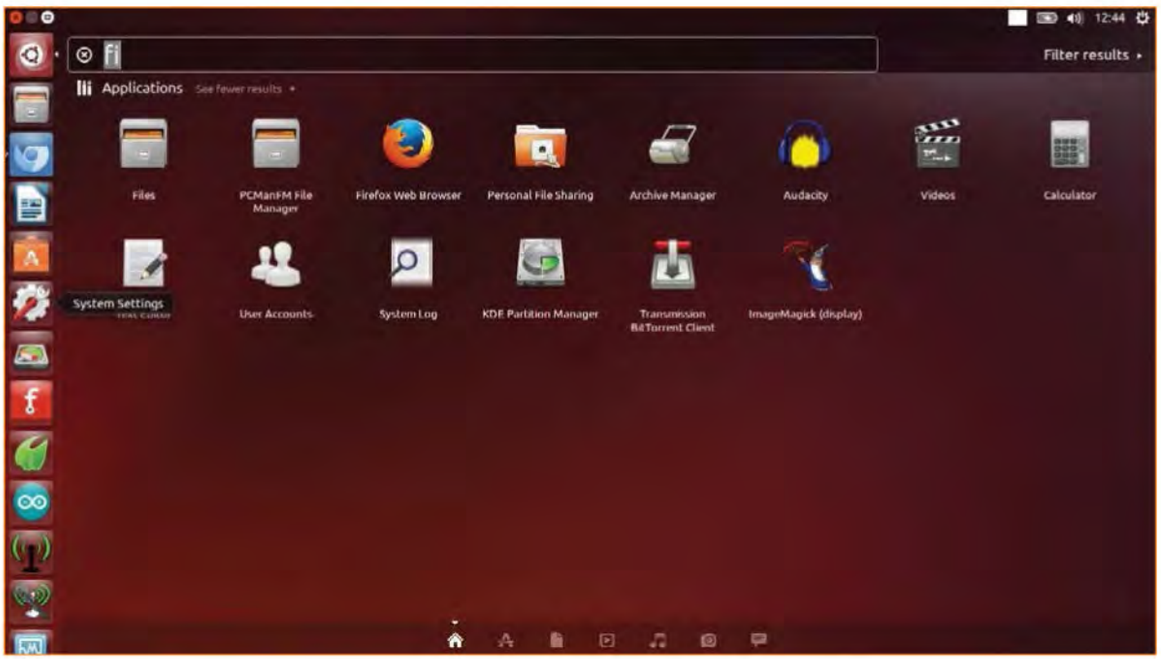
Right now, Ubuntu's focus is split between the desktop and the upcoming Ubuntu phones. Richard Collins, Ubuntu mobile product manager, has been working on the touch OS since he joined Canonical three years ago:

"The idea always came up about what would be the impetus, how would it work, etcetera. It really came from Ubuntu for Android and when we used that product to talk to many mobile manufacturers and mobile operators. The relationships were there and it was a reasonably rational development to start thinking about how Ubuntu as a code base would effectively be unique, in the sense that it could truly operate as a single code base across different form factors.

"It's where the industry was going anyway because you have touch-based laptops and big screens, so the evolution had already started to take place – it just required a strategic push to say 'Right, we're going to be doing the phone as a fully-fledged serious commercial product.'"

Not forgetting the community, Canonical got them very involved in development early on.

"One of the things we wanted to do was get the community involved with the phone, so it wasn't just us producing the phone and putting it out there. We wanted to get people involved. So we started this core apps project, and core apps are the typical applications you need on a phone like a clock or calendar. Then we've got some slightly more esoteric things that you might think are a bit out of place on a phone, like a terminal and a file manager. We have a weather app and a few others as well. All of those apps are developed by people in the community.



Left The latest version of Ubuntu carries on the legacy of Warty Warthog ten years on

Ubuntu timeline

2 January 2013

Ubuntu Touch

At a 2013 press unveiling, Mark Shuttleworth revealed the new Ubuntu mobile OS. The Unity interface had been scaled down to work on touchscreens and showed signs of the unified environment originally promised.

17 April 2014

The current LTS

14.04 is Canonical's most recent Long Term Support version of Ubuntu with five years of support, which appeals to enterprise for both desktops and server. It's a very stable release, focusing on bug fixing and not new features.

“I think we’ve been able to strike a balance between the company and incorporating community input”

“These core apps specifically are done in collaboration with the Canonical design team, so the design team say ‘this is what we think a clock app should look like’ and they provide that design to us.

“We [looked] for people in the community who [wanted] to contribute an app to Ubuntu and sure enough we found a bunch of people who were willing to do so, and they created some of the apps that are going to ship on the many, many phones released over the course of the next year or so. That, for me, is brilliant. They’ll be able to go to the store and buy a phone and their own code is running on that phone.”

“It’s been massive,” Richard confirms the community’s involvement. “The community is embracing everything that we [have done] and announced so fantastically well. A set of applications that are preinstalled on the phone have come from the community and there are hundreds more [that] people can download from the application store.”

Richard also told us of some unspecified future announcements regarding the phone and an established roadmap for its development. What seems clear, though, is that the present and future of Canonical and Ubuntu owes a lot to the active community members who helped it grow. Jane reflected on this when we asked her about working with an open source company:

“I think we’ve been able to strike a balance between the company and incorporating the community input, work and enthusiasm quite well. It’s not always a smooth road – [there are] rocky bits as there are in any relationship or any company. I think it’s one of the most interesting bits about Ubuntu as a distro we continue even as we move into the phone and tablet world, and cloud on the server side. I think we continue to maintain a level of transparency and participation that’s unique.”

Above The apps are largely built by the community in conjunction with a proper design team

Above Ubuntu Touch’s development is tied to Ubuntu for Android, the project for creating a dockable Ubuntu-running phone

A message from Jane Silber

“I personally, and we at Canonical, would like to thank your readers and people in the Ubuntu community who’ve participated, worked and celebrated with us over the last 11 years. Clearly we wouldn’t be here without their contributions, and I know part of your audience [are] developers. Devs in particular continue to be a very important audience for us. We hope we continue to bring them the best development platforms and development tools from Ubuntu itself; from Juju to the ability to spin up Ubuntu images easily in the cloud, we value that developer audience and hope we can give them the best tools possible for the next ten years as well.”

3 July 2014

Goodbye Ubuntu One

After five years Ubuntu One is finally shut down. Due to the changing state of computing, Canonical felt it was no longer an effective service for their needs. Partnerships are the way forward for these kinds of applications.

23 October 2014

Ubuntu is 10

Coming out exactly ten years and three days after the original Ubuntu, 14.10 will continue the Warty Warthog’s legacy of an easy-to-use Linux distribution, albeit now with a more unique flair thanks to Unity.

Get started with Ubuntu Phone

Learn how to set up and get the most out of your brand new Ubuntu phone

You've got your shiny new Ubuntu phone. Perhaps it's the Aquaris E4.5 (covered over the following pages), but it may well be something else in the future – and you want to get it set up and working. If you've ever used an Android phone, the initial quick-start wizard will be fairly familiar to you, and even if you haven't, it's very straightforward.

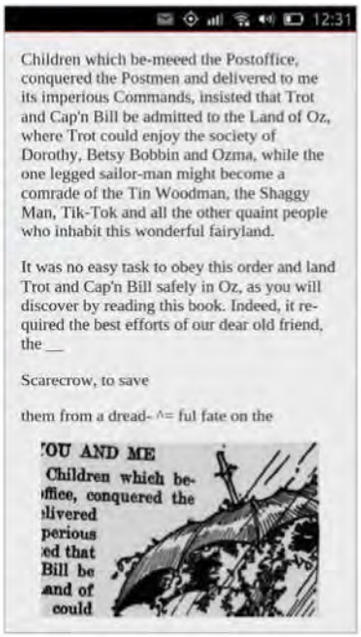
Luckily the phone then shows you around, giving you tips on which way to swipe when actually using the phone – from the left for the launch bar, from the right to move between apps and from the top to access settings – all the stuff that's been talked about many times before.

Account integration

Before doing anything else, it's a good time to start linking various accounts to your phone. If you've owned an Android phone before this and had it linked to a Google Account, your first port of call should be to add this Google Account to your phone. In doing so, you get access to email notifications along with the ability to then sync your contacts listed on the account. It won't do the latter by default, we found, so you may need to go into the account settings to get them.

This account – along with the account information you can set up for Twitter, Facebook and several other social media and related account types – then links to the apps for these specific accounts in the Ubuntu phone system. These link to the web versions of such services, but they enable you to always be logged in and to get notifications from some of them.

Left You can reveal the familiar Ubuntu Launcher by simply swiping from the left edge of the screen



> The home menu gives you access to a series of scopes. From here you can pull up the settings from the bottom

Settings

Once accounts are set, you should look into which settings to change. First, check to see if your phone is able to automatically set the correct time from location, or if you need to set it manually – we were left a little confused on the morning that the clocks went forward, as we found our phone was still an hour behind.

The home menu gives you access to a series of scopes. From here you can pull up the settings from the bottom to add or remove from a series of installed scopes, and any others you install from the Ubuntu Store later on.

Everything else

There are a few extra little things you should be aware of. For example, volume is done with a master setting, so make sure that it is set correctly for your needs. From the volume part on the top bar, you can also hit a Silent profile if you feel it is needed. The difference between swiping from right to left inside an app, swiping to the next app and opening up a view of all open apps depends on where you start swiping (for in-app swipes, stay away from the edge) and where you finish (swipe as far as you can for the overview). There are a few other little quirks that you may or may not notice, depending on your phone background.

Top left New apps and scopes for the Ubuntu phone are freely available to download inside the Ubuntu Store

Top right Ubuntu Store offerings are currently modest but will hopefully expand quickly now that the phone is finally out

Starter apps

What should you look out for to enjoy your new phone with

Beru READING

Kindle functionality is still not available for Ubuntu Phones, but if you do want to read something then Beru has you covered. It uses the same ePub files that other ebook readers use, so you can easily load it up with any of your existing books.

If you don't have any ebooks like that lying around, it lets you access websites with free, public domain books to download to your phone and start reading. Like all good book reading apps, it saves your place and you can customise the text colour to create a better reading environment.

Document Viewer DOCUMENT VIEWING

Document Viewer is part of the Ubuntu Touch 'Core App' set of software, although it's not installed to the phone by default. Getting it is a good idea, and a great way to make sure you can look at documents you get via email or elsewhere. You can't actually edit or create new documents, but it's a good stop-gap until something with a bit more power makes its way to the device. It can view PDFs and a few other document types, and as it's part of the Core App bundle, it will definitely get upgrades in the future.

Google Places scope NAVIGATION

This is a scope rather than a proper app, but it works very well. It uses your location taken from Wi-Fi, cell tower or GPS, depending on what you've got set up, and it will automatically display some info on places in the surrounding area. It's presented very nicely as well, and not like a repurposed web page either.

You can drill down into the results to look for something more relevant to your current circumstances, such as restaurants if you're hungry. It also grabs the extra info you need to find these locales and even contact them if need be.

Dekko EMAIL

The default email apps on Touch are currently only for third-party web services, like Gmail and Yahoo Mail, etc. Dekko gives you a more traditional email client solution that can hook into different IMAP and POP email accounts, with emails then stored locally which you can reply to, forward on and everything else you'd want from an email client. There are a couple of caveats at the moment: there are no push notifications, unfortunately – but then most traditional mail clients don't do that anyway – and it doesn't work with Microsoft Exchange yet.

PHONE

Bq Aquaris E4.5 Ubuntu Edition

Does the first handset to feature Ubuntu for Phones fill the void left by the Ubuntu Edge project, or is it a damp squib?

Originally created as a low-end Android handset for the Spanish market, the Bq Aquaris E4.5 has been retooled for Canonical's Ubuntu. The only visible change: the removal of the soft buttons around the low-resolution qHD display, dropped in favour of Ubuntu's Sailfish-inspired, edge-swipe navigation system. Everything else is identical, right down to the lightweight plastic chassis, which feels undeniably cheap in the hand, and the buggy interpolated camera that fails to live up to the eight-megapixel claims of its manufacturer.

The hardware isn't the star of this show, of course, and that's just as well. The Aquaris E4.5 Ubuntu Edition's selling point is its operating system, a variant of Ubuntu for Phones based on the release of Ubuntu 15.04.

Operating System
Ubuntu for Phones 15.04

Processor
MediaTek MT6582 Quad-Core ARM Cortex-A7, 1.3GHz

Memory
11GB RAM, 8GB Flash Storage (Upgradeable via Micro-SD)

Dimensions
137mm x 67mm x 9mm

Display size
4.5"

Display resolution
540 x 960 (qHD)

Wireless
Bluetooth 4.0, 802.11b/g/n Wi-Fi (2.4GHz only), GSM 850, 900, 1800, 1900, UMTS 900, 2110

Price
€169.90

> We frequently found the handset entering a state where the pop-up keyboard would refuse to appear

Sadly, the features Canonical promised as part of its failed Ubuntu Edge crowd-funding project have yet to be realised. There's no support for a desktop interface and you can't connect an external display. Instead, you get an operating system which feels remarkably like an early build of Android – right down to the icons used in the overly-busy and cramped notification bar, which pulls down to reveal a notification and quick-settings centre just like its mainstream inspiration.

That is not to say that Ubuntu for Phones is a simple Android clone, however. It plunders numerous rival platforms for inspiration, from MeeGo and Sailfish through to iOS, all in support of the operating system's central pillar: Scopes.

A Scope, in Ubuntu for Phones parlance, is designed to pull information from multiple sources into a single screen for at-a-glance viewing. The default Scope, Today, shows current and upcoming weather, calendar entries and recent calls and messages. A swipe to the side switches to the NearBy Scope, which duplicates the weather information and uses your location to search Yelp for landmarks around you.

The idea of a Scope is neat enough, but its current implementation feels lacking and forced, while the duplication of data between Scopes leaves you wondering if it's enough of a gimmick on which to hang an entire mobile platform. Lacking, in fact, is a particularly good word to describe the operating system as a whole.

Ubuntu on the desktop might be stable but Ubuntu for Phones needs a bit more time in the oven. During our test, we frequently found the handset entering a state where the pop-up keyboard would refuse to appear, while Wi-Fi – already limited to 2.4GHz networks as a cost-cutting measure – would cease working altogether until the handset was rebooted.

The Ubuntu Store provides add-on software, but its selections are sparse. Few brand-name packages are available, and those that do appear are typically little more than a shortcut to the company's mobile website. With no compatibility layer for running Android applications, that makes using the Ubuntu Edition as anything other than a secondary handset a limiting experience.

Using Ubuntu on the Aquaris also opens up the question of missed opportunities. Using the handset requires an Ubuntu One account, but with Canonical having closed its cloud storage and content synchronisation service back in 2014, it has nothing to compete with Google Drive or Apple iCloud. Had Ubuntu One survived in anything but name only, perhaps using Ubuntu on a smartphone would be a more pleasurable experience; as it is, the overwhelming feeling is one of disappointment.

Pros

A low purchase price makes the Aquaris E4.5 an affordable second handset, and the dual-SIM functionality is handy

Cons

The software simply isn't ready for prime-time use and the ecosystem is barren, while the cut-price nature of the handset is easy to see

Summary

Cheap, low-end hardware and a buggy operating system make the Bq Aquaris E4.5 Ubuntu Edition tough to recommend, while its principle ideas have been implemented better elsewhere. Ubuntu for Phones is definitely a project to keep an eye on, but not one on which to spend your hard-earned cash just yet.

Below Test images are available now and the aim is to have the full system ready for summer 2016

INTERVIEW SEBASTIAN KÜGLER

Plasma Mobile: next-gen Linux phone

Sebastian Kügler
is a key developer and former board member of the KDE team. He works for Blue Systems, which is the main driver behind Kubuntu, creator of Netrunner and a major contributor to the KDE project.

Core KDE developer Sebastian Kügler reveals how he is heading up the project that is bringing the Plasma desktop to phones, pushing Linux further into the mobile market

Congratulations on recently making the project announcement! How long have you been working on Plasma Mobile now?
In earnest, about four to five months. It started as a very proof-of-concept thing about a year ago, and around the change of year we knew that we could actually bring up a Plasma UI on a phone. So then we started to figure out the stack – how things should work together, how to get it working on the Nexus hardware. We tried to move to the Ubuntu-based stack but had huge problems getting libhybris to work: kernel and graphics driver problems. That actually threw us back about two to three months and it was basically solved by moving to KWin completely. Then the guy who does development on KWin could actually get it to work and from there we could start on – well, actually race through – development, so we would have something that we would actually be proud of showing at the Akademy conference.

So you are now based on the Ubuntu stack?
Yes. The idea is that we create the whole software environment, but that you can actually, to a certain degree, exchange the underlying operating system – much like a Linux desktop, basically. So you have a model operating system, such as Ubuntu Touch or Mer, Sailfish, and you make sure the KDE frameworks are packaged and that we can use a graphics interface – either DRM or libhybris. Then on top of that, everything else should stay the same.

Your website talks of this being inclusive in terms of its app support – running the Plasma apps, Ubuntu Touch apps, GNOME and X11 apps. What's the status with the Android apps?
So it's technically possible but it's not easy, because there's a lot of infrastructure needed to emulate a complete Android environment. There's this one guy in our team working on it full-time, who gets help left and

Below The aim is to create a coherent KDE experience across both desktop computers and mobile devices

right from others. The big thing, but apparently one of the last really big blockers, is graphics again. The story of our lives: we struggle with graphics all the time! On the desktop we have kind of sorted it out, but now on all these mobile devices, it's a headache all over again.

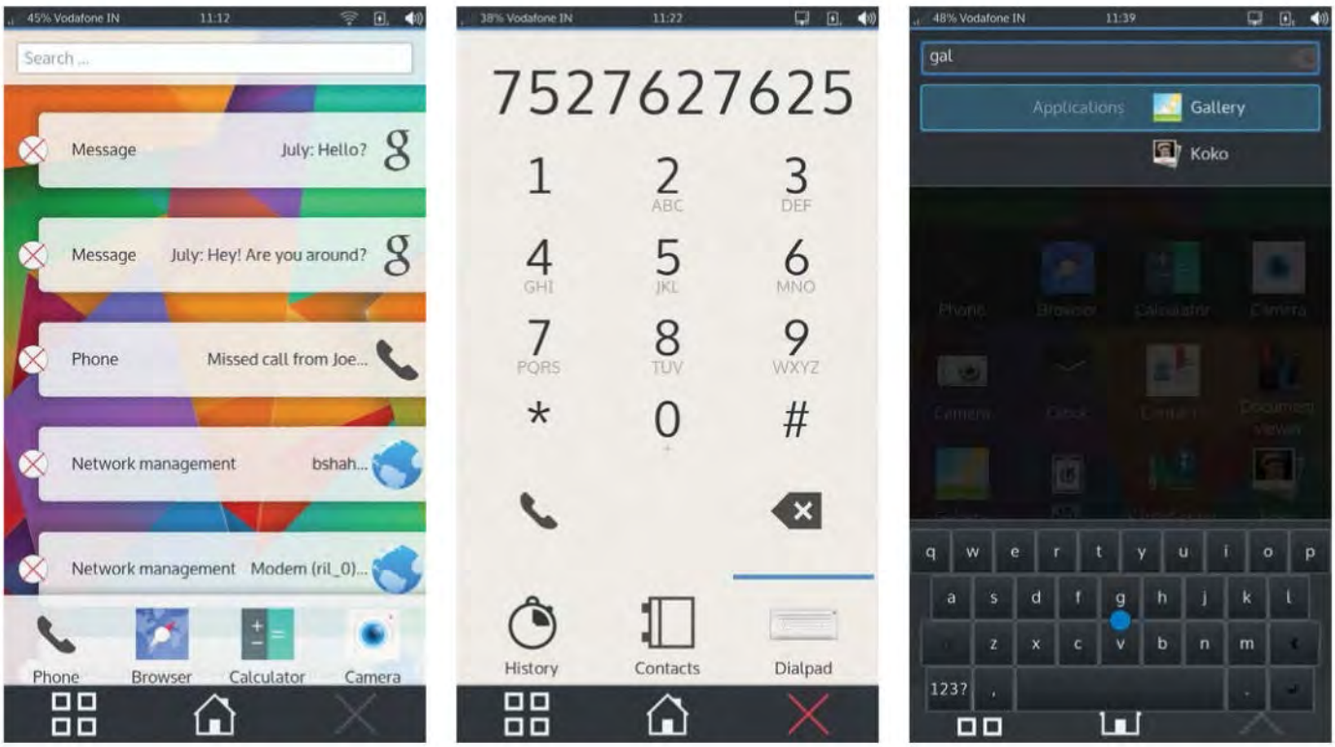
So the status is that he was actually able to start an Android application, but not output graphics yet. It's a milestone for a developer and completely useless to a user right now; there's this huge pile of work that needs to happen before anything actually gets drawn on the screen. With the Android graphics infrastructure there's an interface called SurfaceFlinger that is semantically different from how we do graphics, so there's some kind of bridging needed. The applications actually need to think they're rendering to SurfaceFlinger, but we make them render through a SurfaceFlinger which is backed by Wayland. It's a bit tricky, but we have something that can draw on the screen. At the Akademy conference, we had a very exciting black rectangle.

In terms of the development, is this your current focus – working on display for the various apps – or are you still focusing a lot on the core system itself?

The biggest task right now is actually nontechnical; so we've just put this project out in the open and one thing I've been especially busy with in the past weeks is the project management side. We've seen a lot of excitement from people. Some were very critical, saying 'the world doesn't need it' and 'it doesn't look good'. But there were a lot of people who thought this was a really good idea, something worthwhile – an operating system that is not controlled by a single company, which is developed in the open, that they can actually help to shape. Now we actually have to scale the processes behind it up. So right now we're introducing a project management tool called Phabricator that allows us to coordinate. It's very good for task management and it allows code review. I've also been pretty busy just writing documentation, which is one of the big scalability things – we need the documentation in order to be able to attract developers.

> "An operating system that is not controlled by a single company, which is developed in the open, that people can actually help to shape"

What I really like is that we've presented a prototype with near zero design work in it and we got a lot of feedback of the sort, 'But it looks like shit – they should put a designer on that'. And fair enough, they're completely right. We know that we need to one-up on the design front. But for every ten people who made a critical remark from their armchairs, we actually had people who were sending in mockups for things, thinking of how to improve it. We have already attracted four or five designers who have done real professional work on different aspects of the system, so we're now also



Above left Notifications are at the top of the interface. You scroll down to the apps

Middle The phone app is fully functional and well integrated with the contacts

Right The powerful KRunner launcher has been ported to Plasma Mobile, too

Volunteers vs staff

Sebastian explains the necessity of having people working full-time:

"There's a lot of stuff that you can't really do as a volunteer, where you may have just two hours: they need structural work, they need someone sitting down everyday looking at issues, fixing the running systems. Our code is being built and tested every day – right now we create two or three new images every week, so we can always test the latest version. I know that if I push new code now, then it can be in the next flashable image tomorrow morning. We wanted to get these cycles of development, feedback and testing really, really short."

working on getting them integrated in the community and defining workflows.

How many people are there in the core team – is it quite small compared to some of the other projects under the KDE wing?

Actually, it's not that small. My employer, Blue Systems, is investing in this, so as opposed to most other projects we have quite a strong development team, not only in the number of people but also the time that people can put into it. For example, I'm pretty much full-time on this project, and so are about ten to twelve other people, and that is really necessary to get it off the ground to fill in the basic functionality.

Is this a completely new and independent project, or has there been much engagement with the previous Plasma Active project?

Plasma Active was mine and a few other people's baby. So in a way, Plasma Mobile is a continuation of the Plasma Active project, but with slightly different goals. This time around, we're much more down-to-earth. We're not so much focusing on a novel user experience, but we're trying to get lots of different applications running, which I think is a bit more realistic, at least to fill in the basic functionality. We don't yet know where exactly, design-wise and workflow-wise, we'll end up because it's an open process – open for participation and, with that very nature, also very open-ended. But I think there's more manpower behind it now, more of a sense of urgency within the KDE community to not only do desktop stuff, but also to finally be a project for people who want to get their applications running on mobile as well. It's not about UI paradigms and creating the next big thing – we want to do a bread-and-butter thing first.

There are definitely some unique aspects to Plasma Mobile – replacing the idea of swiping between homescreens with the single, endlessly scrolling page, for example, and incorporating Plasmoids.

It's actually a lot of fun because by reusing the infrastructure we already have on the desktop – in terms of theming, Plasmoids – we can pump out cool functionality pretty quickly. For example, the theme-switching settings application was, I think, about two to three hours of work. Sometimes it's a bit more involved, but we actually have a whole lot of very complex applications that have seen bug fixes for years and are very mature and stable. And by just putting a different UI on top of them we'll be able to get them running on mobile.

Do you have a roadmap for the development ahead?

We want to have an end-user-bearable version next summer. Something that doesn't have real show-stoppers, that could be used to make phone calls, that I could use as a daily driver without going nuts about lack of quality. Last week we sat together and the user experience people said, 'Before we come to any

Vision statement

Sebastian and the Plasma Mobile team recently set out their project vision for their mobile OS:

"Plasma Mobile aims to become a complete software system for mobile devices. It is designed to give privacy-aware users back the full control of their information and communication. Plasma Mobile takes a pragmatic approach and is inclusive to third-party software, allowing the user to choose which applications and services to use. It provides a seamless experience across multiple devices. Plasma Mobile implements open standards and it is developed in a transparent process that is open for the community to participate in."

Left Akademy 2015, where Plasma Mobile was announced, was held on 25-31 July in A Coruña, Galicia

meaningful design, we need to know what the project vision is: we need to think about target groups, otherwise we design something to look nice, but we don't have a clear idea who it's for'. So we took a few steps back and started out with a project vision (see the Vision Statement boxout above).

We're not at a point right now where we can, with a straight face, say this is good for your privacy, but we set the direction in order to have it be very privacy-aware, because we think it's really our mandate as a free software community to give users the tools that also protect their privacy. The free software community has traditionally been very good at that, but we have left the mobile market pretty much to Google and Apple, and they're just not parties to be trusted – with all due respect. Google's business model is… something with what once was your privacy. I want to be able to go to my government and tell them, 'You should use this, and then your chancellor can actually have a private call with the prime minister.' I want to be able to give it to the journalist who isn't sure how to communicate with their peers or informants. And for that we need to create a complete system which can be audited, which is open source, which you can have implemented by someone you trust, which supports strong encryption for email, all these kind of things. So that's why we put that explicitly in the project vision: because we think this is missing right now and it matches very well with the pedigree of a free software project.

> "We want to have an end-user-bearable version next summer – something that I could use as a daily driver without going nuts"

Will this be a download for users to flash themselves, or are you considering hardware options – for example, partnering with Canonical, Bq and Meizu, or looking at older projects such as the Improv or Vivaldi?
For now we're concentrating on the software, but I don't think we live in an age where we can tell you, 'Download this image and flash it onto your device'. That will not reach the mass markets. It may be fine for developers, may be fine for some power users, but I think in order to truly succeed, we'll need to get this preinstalled on phones at some point. We have been in contact, for example, with the Fairphone people, who created a telephone based on sustainable values. There have been some emails exchanged, but given that the software is not yet ready, it is still a bit too early to think about that. But yeah, the idea is definitely to have it preinstalled but also to have it well-documented, with a nicely set-up software stack.

There are smartphone makers out there who are not happy with the dominance of Google and the role that Google plays in their internal strategy: someone else decides what their next software on your phone will be and you'll only get to see it once it's done. In a market where hardware becomes cheaper and cheaper, and in many cases less of a unique selling point, these makers need ready-to-go smartphone systems to install on their devices. And just using Android is right now the easiest, and pretty much the only, option for them. We want to give them more choice in that.

THE FUTURE IS OPEN SOURCE

LinuxCon Europe 2015 saw even more announcements from Linux Foundation Collaborative Projects and Workgroups – we asked Jim Zemlin to explain the strategy driving them

Shortly before LinuxCon Europe kicked off, the Linux Foundation published a report stating that its Collaborative Projects have added an estimated five billion dollars of value to the technology sector. It is a fairly staggering amount and provides real proof of the value of collaborative development – a proof that is more important than ever at a time when companies are increasingly switching to open source development models, abandoning the old internal research and development structures in favour of outsourcing aspects of this to the Linux community. The Linux Foundation calls this concept ‘distributed genius’ and believes that only the distributed genius of thousands of people working as one can solve the most challenging problems of our time. We spoke to Jim Zemlin to hear from him just why the future of the technology sector is in open source.

“Why don’t I just start with the broader perspective in terms of what’s going on here?” began Jim. “This report is one more example – I would argue amongst a really broad trend that’s happening in the software industry – where open source software is really becoming the way software gets developed. And what I mean by that is that the percentage of open source code in *any* technology continues to increase month by month, year by year, to the point where companies are, essentially, collectively creating the underlying components that they use to run the London stock exchange, Amazon, Facebook, Google, Android devices, Samsung televisions, Qualcomm software and so forth – that has become a permanent part of the way software is being developed. And what all those companies do is instead focus on their financial trading algorithms, government compliance, their customers, their trading platforms; all that underlying code is open source and you share the development of that with your peers. So this report answers the question very concretely as to why. I think that a lot of people recognise that open source is a better, faster, cheaper way to build software, that many people are smarter than any one person, but this really puts a fine point on it by quantifying just how much value is being created in these projects and in the Linux Foundation projects.”

Jim Zemlin
is the executive director of the Linux Foundation. His career spans three of the largest technology trends to rise over the last decade: mobile computing, cloud computing and open source software. Today he uses this experience to accelerate innovation in technology

According to the published report, this quantified value is over five billion dollars. “To put it in even more context,” Jim continued, “you could also go to GitHub, you could go to SourceForge and other places, and there’s even more code worth tens of billions of dollars – and remember, this doesn’t even include Linux, which is worth ten billion more. So the message here is that if you look at these big centres of open source development, it is billions and billions and billions of dollars of value being created and collectively shared every single day. This is why the nature of software development has shifted; companies have said, ‘Hey, I’m not going to compete with that! Why would I spend internal research and development dollars?’ The top ten tech companies in the world spend 64 billion dollars a year on R&D. What they’re now saying is, ‘I’m going to shed some of that R&D out into open

Right The Foundation's role as mediator and facilitator helps it create neutral spaces for crucial development work, such as on the kernel itself

FOSSology hosting

Created by HP in 2007, FOSSology is an open source license compliance software project. It takes a software package and analyses each of its constituent files, using a heuristic pattern-matching algorithm to identify and report all of the software licenses. HP created it in order to quickly and accurately evaluate open source software that was proposed for use within the company, as well as software being considered for distribution. The company open-sourced the project to share its benefits with other organisations.

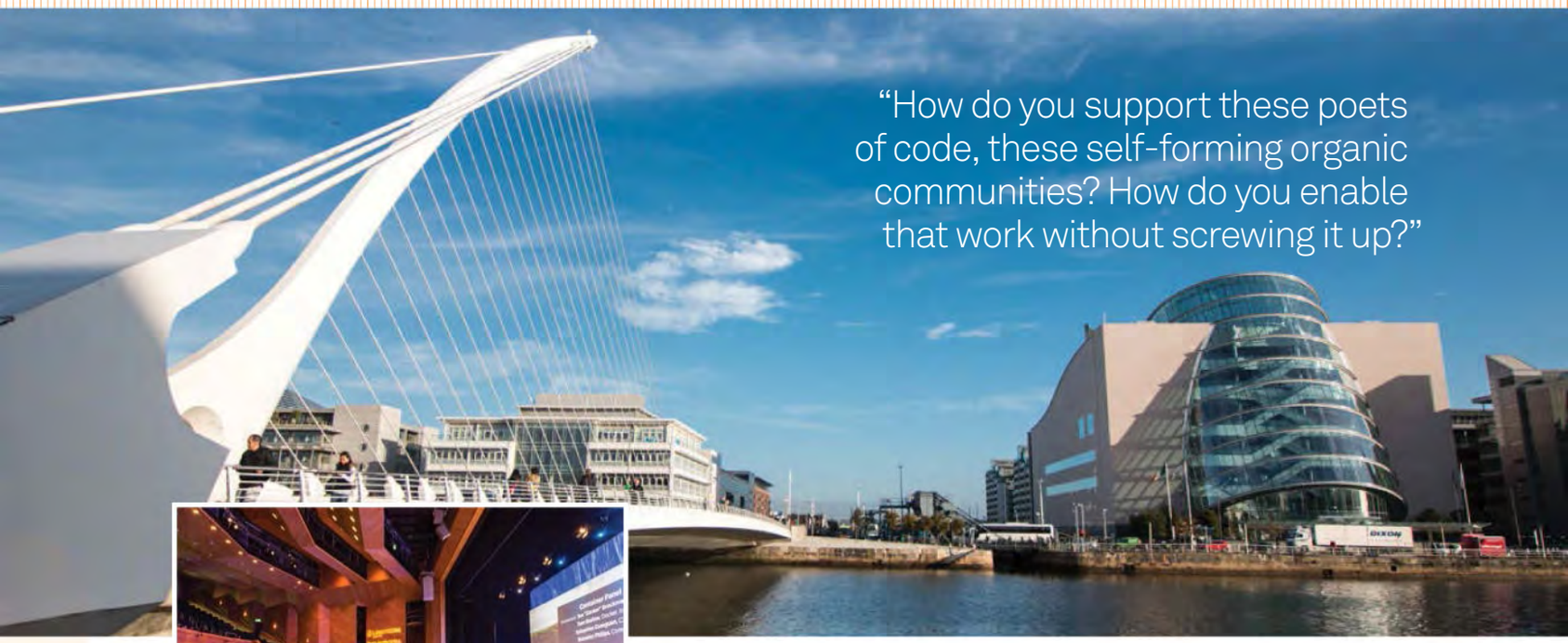
FOSSology is used by companies ranging from Siemens to Intel's Wind River in order to run license and copyright scans in one click, and generate Software Packet Data Exchange (SPDX) and Readme files with the copyright notices. In need of project hosting, asset ownership and project leadership, FOSSology investigated potential backers earlier this year and is now hosted by the Foundation.

source and instead I'm going to focus my internal R&D on the things that truly matter to my customers and truly make me unique.' "

Dronecode is an excellent example of this. The UAV (unmanned aerial vehicle) marketplace is expanding incredibly quickly and 90 per cent of UAVs are powered by the Dronecode code base. As Jim put it, if you're going to start a UAV company in precision agriculture or mining, why would you build an entirely new operating system when you can get the entire set of software components for free? We asked Jim what he saw as the turning point in the industry and whether he could identify catalysts that gave momentum to this movement towards open source development models.

"Linux really is the existence proof," explained Jim, "that through collective innovation and collaborative development, you can have shared R&D that actually enables business. In other words, it's not some kind of socialist experiment here. Linux proved that Red Hat can be a ten billion dollar company; that Amazon, Facebook, Google, most of the Internet, can run on top of free software; that every single major stock market in the world can run on top of free software. That really was a huge catalyst for people to go, 'Holy cow – that is working!' Then the mobile industry started growing – another huge sea change in computing. You hear all CEOs talking about how the move has been from web to mobile. Well, guess what powers mobile computing? Every Android device is a Linux device, right? 80 per cent of the software in most Android phones is open source software, and that enabled another wave, an existence proof saying, 'Wow – this stuff is really working!' And then we get into cloud computing. If you look at the vanguards of cloud computing, essentially software-as-a-service – Amazon, Google, Rackspace with OpenStack, IBM's cloud – all are powered by Linux and open source projects like Kubernetes, Cloud Foundry and others. The cloud is essentially built on open source.

"So now you're starting to get this snowball effect where the turning points keep accelerating. It's like, 'We know we can do this; Linux worked. Let's apply it to mobile. Let's apply it to cloud. Let's apply it to big data with Hadoop.' One of my favourite turning points is that the CEO of Microsoft, Satya Nadella – whom I really respect – came out and said, 'Hey, we love Linux! We want Linux and open source software to run well on our Azure cloud.' Microsoft participates in many of our projects: the Open Container Project, the Node.js Foundation, our Core Infrastructure Initiative. I think that's a big turning point where one of the most important companies in the software sector, Microsoft, is now a first-class citizen in open source."



"How do you support these poets of code, these self-forming organic communities? How do you enable that work without screwing it up?"

Above The annual convention unites Linux talent from across the globe in order to share knowledge and solve problems

The Linux Foundation's role as a cohesive force that enables collaboration is more important than ever. As well as its overarching Collaborative Projects, the Foundation has also announced specialised workgroups that are solving specific problems and providing tools that feed back into this wider collaborative effort. Often these are pre-existing projects that have been left to languish through a lack of support or funding, and the Foundation has effectively rescued them. As announced at LinuxCon Europe, for example, the FOSSology project is now being hosted by the Foundation having sought backing for a number of years, and the Real-Time Linux workgroup has been set up to formalise and protect critical work that was being done by one person in their spare time.

We asked Jim whether projects like this are being accepted under the wing of the Foundation as and when they've asked for help or whether the Foundation actively selects projects.

"It is the latter," Jim replied. "What you're seeing is that as open source becomes more part and parcel of how software gets developed, you need a set of structures around that which are more sophisticated to facilitate the flow of that code. Now FOSSology, as an example, facilitates the flow of code by making license compliances. Real-Time Linux is the kind of technology that runs aircraft carriers, the Mars Rover – stuff that is incredibly important to people's lives. In some cases, we've found that there are either specialised areas of expertise that have gone unfunded or projects that for a variety of reasons have not gotten the resources commensurate with their role and tech, and we seek to remedy those. We did this with our Core Infrastructure Initiative, to go and underwrite the OpenSSL project after Heartbleed. We looked at that and were like, 'Hey, the Internet can't be secured by two guys named Steve who work part-time. These people need to have resources and assistance!' We provided that for them and we're doing the same thing here with Real-Time Linux. Thomas Gleixner and the Real-Time kernel that he's been developing is extraordinarily valuable, very specialised knowledge, and the Linux Foundation can play a hand in getting resources to those developers so that it can better the world in terms of how that technology impacts society. Now the key here is that we want to see that Real-Time work in the mainline Linux kernel. We want to see it land in a place where it will be maintained over a long period of time in a robust, vibrant, living community; that's the goal. And Thomas Gleixner is one of the smartest people in his field – I would argue that he's perhaps *the* smartest person in his field!"

So the Foundation is not only helping people to collaborate with each other, but it is also safeguarding particular roles so that the knowledge held by people such as Thomas Gleixner and the two Steves (Marquess and Henson) won't disappear. Moreover, it is so that the development work done in these contexts can then be fed back into the mainline for the betterment of all. It is a delicate task. "We are like the janitors at the Linux Foundation," added Jim. "We make sure that the

OpenChain Workgroup

The Linux Foundation's OpenChain Workgroup is a community effort to standardise common best practises and ease open source compliance for companies and developers. It is expected to reduce costs and the duplication of efforts as well as ease the friction points in the software supply chain. As Jim put it, "Because nearly every new technology today is built using Linux and open source software, today's software supply chain is the open source software supply chain. This means we need to revisit the way we standardise processes and compliance for checking code and ensure the cost and efficiency benefits of open source are sustained for decades to come."

What this means is that the OpenChain Workgroup will work to provide a baseline process that can then be customised as companies and developers see fit. This will initially require providing a set of guidelines that will then be intended to be used as a basis for monitoring and developing compliance training. This should leverage existing best practices within the Linux ecosystem.

Right Containers are another area where industry-wide collaboration is yielding incredible results

Real-Time Linux

Real-Time applications have operational deadlines between the triggering of an event and the application's response to that event which must be met. This enables a guaranteed response time – usually for safety reasons, as with pacemakers. Real-Time Linux is a kernel patch set (CONFIG_PREEMPT_RT) that makes Linux into a real-time system, enabling the control of robots, data acquisition systems, manufacturing plants and other time-sensitive equipment and machines from RTL applications. It's also important because of the number of bugs in the mainline kernel that it identifies and fixes as part of the patch development.

The future of Real-Time Linux has been under threat because of a lack of funding, with the core maintainer Thomas Gleixner – who has worked on the patch set for over a decade – being reduced to working on a hobbyist basis because of a lack of support. As announced at LinuxCon Europe in October, the project is now officially backed by the Linux Foundation and has become a Workgroup, with Thomas Gleixner being made a fellow of the Foundation so that he can dedicate more of his time to working on the project.

Above Virtual reality is as exciting for Linux developers as it is for others, with virtual reality a talking point at LinuxCon

bathrooms are clean, that the facilities are wonderful and the code can flow, but what we don't want to do is screw things up! The professors and students and academics who create the real value here are the developers. The key is, how do you support these poets of code, these self-forming organic communities? How do you enable that work without screwing it up? Linus was really impressive when he had the foresight years ago to say, 'We don't have a plan for Linux. We're going to let that technology adapt to modern computing needs as they evolve in a somewhat organic fashion.' And that's the attitude we take as well. We're the supporting cast here. We're the roadies – the rockstars are the coders and we're the guys that make sure that the lighting is really good, that ticket sales happen on time."

It's a neat metaphor, but how does it translate in a day-to-day sense? Is the Foundation just providing the neutral spaces and infrastructure for these projects and workgroups, or does it take more of an active role within these projects, shaping budgets and guiding decision-making? "It's both," answered Jim. "The key is that there does need to be a neutral place to house the intellectual property assets for these projects, so that people have confidence that no single entity controls them. That's a core tenet to what we do: Linux is neutral because it's collectively owned; Linus works for us in a neutral capacity. Node.js is collectively owned at the Node.js Foundation. Cloud Foundry is collectively owned at the Cloud Foundry Foundation. We want to provide all of the infrastructure around that to make that enablement easier – so what is the best development infrastructure we can provide? How do we help set meetings and get the people who make the decisions on this technology together? How do we host events? How can we provide training? For example, in Cloud Foundry they work and do para-programming dojos where developers come in and, essentially, program side-by-side for a six-week period in order to on-ramp for that project. So we're providing all of that infrastructure.

"We do provide guidance," continued Jim, "in as much as a neutral third party brokering an agreement amongst other parties provides guidance. We're not a judge, we're a facilitator. We're not making these decisions, the

communities make those decisions – we just help those communities make those decisions as efficiently as possible. And again, that's where my biggest fear is: not that we do a bad job on all the infrastructure and the intellectual property sharing components of it, it's that we screw up by trying to guide decisions that are really meant for developers to make. That's why we have a very clear mandate in our organisation that we don't make those decisions."

Another huge benefit of the Foundation's efforts towards facilitating industry-wide collaboration is that the projects it individually nurtures drive the development of components that improve the effectiveness of open source as a whole. "If you think about these projects as horizontal lines up and down the stack – Node.js at the framework layer, Cloud Foundry at the PaaS layer, Linux at the operating system layer, networking and Open Daylight below that – we're going to see lots more of those projects up and down the computing stack. What's interesting to me are the vertical lines that *intersect* each of those projects, where the Linux Foundation can offer *scale* to each of those projects: governance, IP management, license compliance, security best practises, training, developer certification programmes. These are things that every project needs, and most open source projects have a really hard time doing any of that stuff on their own – they don't have attorneys on staff, they don't have training experts on staff, they don't have people who are experts in building open source communities on staff. But the Linux Foundation does, and so weirdly we help projects collaborate but we also help collaborative projects collaborate amongst themselves.

"I need to educate an entire generation on not why to do this collaborative development – they're sold – but specifically *how* to do it. How do I get the next thousand companies who don't even know much about open source at all, but know enough to say, 'Wow, this is a big part of the code base in my product', how do I get them to think strategically about this and participate in a more robust way? Because they're going to – they have to, based on the market. If they don't embrace open source then they're going to have a hard time competing, so anything I can do to get that process to happen faster is good for all of us. That's the challenge that we have and that's why you see us announcing projects like FOSSology and OpenChain – these are essentially things that are making it easier for organisations to consume and redistribute open source code as part of their normal development process. From our perspective, the more open source software that is out there, the better it is for society as a whole – that's what we want to do, that's our beacon."

Value of collaboration

Quantifying the value of an open source project, let alone an entire stable of them, is no simple matter. However, the Foundation has done just that, creating a report based on David Wheeler's COCOMO (Constructive Cost Model). In 2002, Wheeler used the COCOMO, based on the work of Barry Boehm, to analyse a typical Linux distribution, counting the lines of source code and estimating the amount of development effort required to produce it in person-years (the amount of work one developer can do in a year). He then produced a final estimation of the costs.

The Linux Foundation's report, *A $5 Billion Value: Estimating the Total Development Cost of the Linux Foundation's Collaborative Projects*, estimates that there are 115,013,302 lines of source code among its Collaborative Projects. The estimated total amount of effort required to reproduce that code is 41,192.25 person-years. In other words, it would take a team of 1,356 developers 30 years, which equates to a value of over $5 billion.

Enhance your system

044 Build your own system tools

052 Troubleshoot & Repair Linux networks

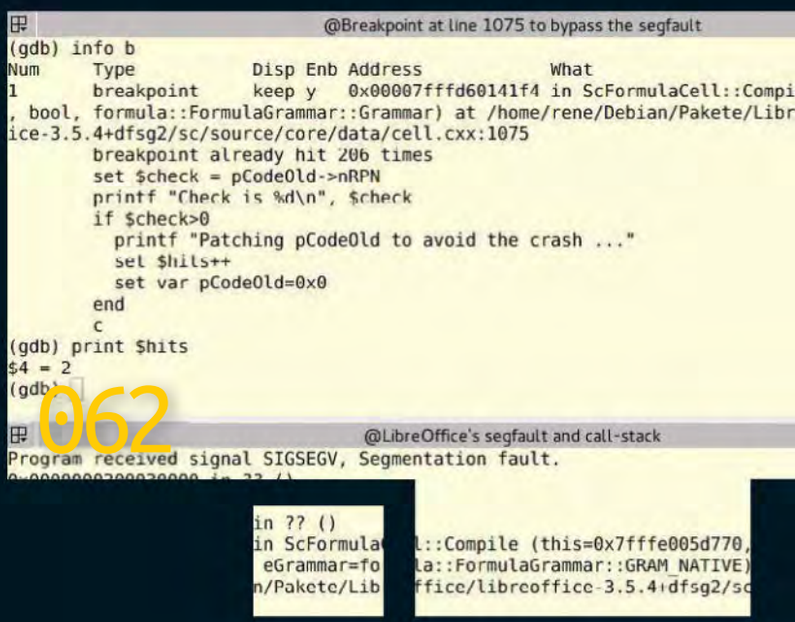
058 Turn an old PC into a NAS box

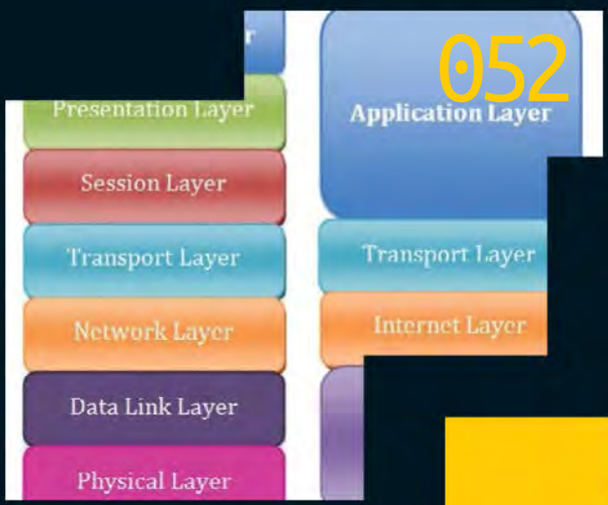
062 Debug your own Linux software like a pro

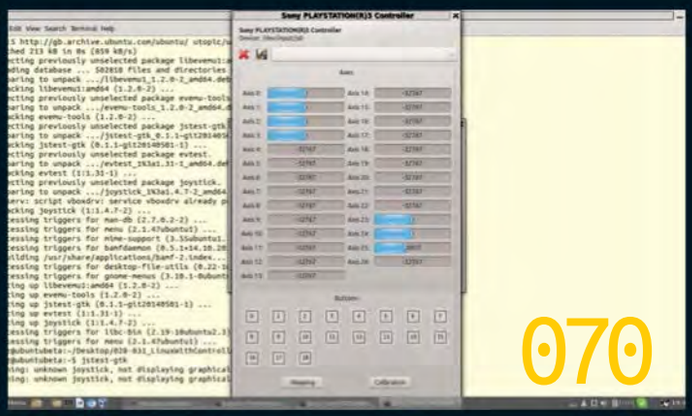
066 Automate your audio in Linux

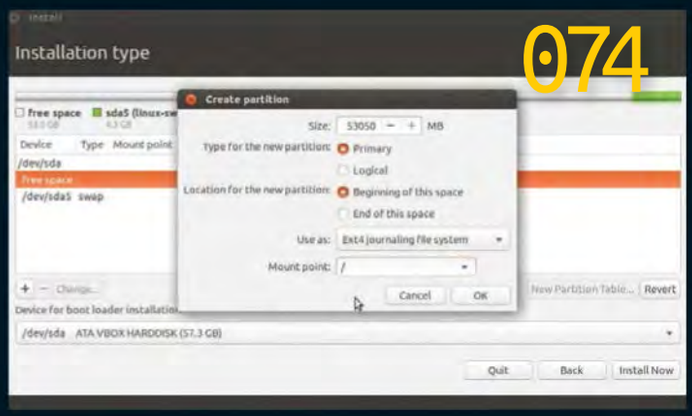
070 Set up game controllers to work on Linux

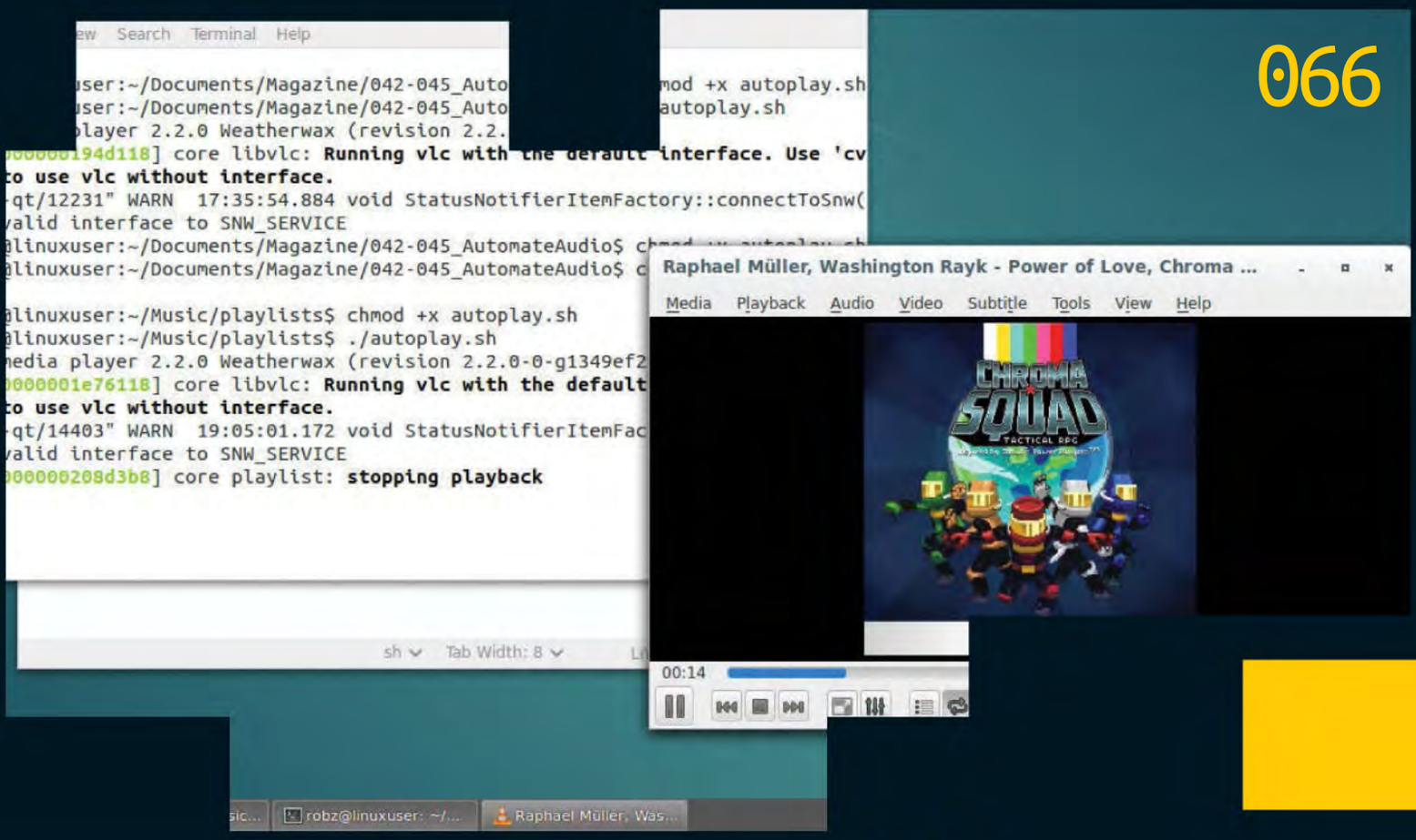
074 Speed up your system by combining SSD and HDD






070


Installation type
074
Create partition
Size: 53050 MB
Type for the new partition: Primary
Logical
Location for the new partition: Beginning of this space
End of this space
Use as: Ext4 journaling file system
Mount point: /
Cancel
OK
/dev/sda ATA VBOX HARDDISK (57.3 GB)
Quit
Back
Install Now


066
Search Terminal Help
user:~/Documents/Magazine/042-045_Auto
user:~/Documents/Magazine/042-045_Auto
player 2.2.0 Weatherwax (revision 2.2.
194d118] core libvlc: Running vlc with the default interface. Use 'cv
to use vlc without interface.
-qt/12231" WARN 17:35:54.884 void StatusNotifierItemFactory::connectToSnw(
valid interface to SNW_SERVICE
@linuxuser:~/Documents/Magazine/042-045_AutomateAudio$ chmod +x autoplay.sh
@linuxuser:~/Documents/Magazine/042-045_AutomateAudio$ c
@linuxuser:~/Music/playlists$ chmod +x autoplay.sh
@linuxuser:~/Music/playlists$ ./autoplay.sh
media player 2.2.0 Weatherwax (revision 2.2.0-0-g1349ef2
0000001e76118] core libvlc: Running vlc with the default
to use vlc without interface.
-qt/14403" WARN 19:05:01.172 void StatusNotifierItemFac
valid interface to SNW_SERVICE
000000208d3b8] core playlist: stopping playback
sh Tab Width: 8
Raphael Müller, Washington Rayk - Power of Love, Chroma ...
Media Playback Audio Video Subtitle Tools View Help
CHROMA SQUAD
TACTICAL RPG
00:14
robz@linuxuser: ~/...
Raphael Müller, Was...

BUILD YOUR OWN SYSTEM TOOLS

MASTER SYSTEM CALLS

UNDERSTAND PROCESSES

WORK WITH C FUNCTIONS

USE MULTIPLE LANGUAGES

32mm

18mm

50mm

14mm

32mm

Harness C system calls and start learning systems programming to develop your own software

We're going to delve into systems programming here, with the aim of helping you to understand how various Linux utilities and servers are implemented as well as what is going on behind the scenes on your Linux system. It would be good to have a basic knowledge of C and know how to compile and run a C program before continuing.

By the end of this systems programming feature, you will be able to develop system utilities and understand the brilliance of Unix that lies in its simplicity and consistency.

It is also very interesting to realise that the principal functionality of Unix system software is relatively easy to implement and does not usually take lots of code. Most of the C code is about error checking, processing command line arguments and making the C code as portable as possible.

WHAT IS SYSTEMS PROGRAMMING?

System calls and techniques of system programming

Linux and Unix

We mention Unix a few times here. For clarity, the UNIX® operating system was created in assembly language by AT&T and later re-implemented in C, also created by AT&T. Unix is now generally accepted as an umbrella term for UNIX® and its derivatives. The Linux kernel is Unix-like, so these systems programming techniques apply to Linux.

Systems programming is a special area of programming in a Unix environment. Back when Unix was first introduced, the only way to write systems software was by using C. Nowadays, you can program systems software using other languages as well.

Most commands that have to do with system administration tasks, such as disk formatting, network interface configuration, module loading, kernel performance tracking and so on, are implemented using the techniques of systems programming.

The key difficulty in systems programming is the fact that a wrong system call can make your Linux machine misbehave or, even worse, crash altogether. Most Linux security issues usually come from such wrongly implemented system software, so you should be extremely careful when using system calls because code with bugs can compromise the security of an entire system. The worst part is that this can happen many years after using the vulnerable software.

Areas of systems programming

It is possible to group system calls and techniques into the following categories:

File I/O: This is the first thing that an operating system will have to deal with. File I/O must be efficient, secure and easy to implement.

Advanced file I/O: Apart from the basic I/O system calls, there is a plethora of additional calls that offer nonblocking I/O, asynchronous I/O and so on.

Files and directories: Besides reading and writing to regular files, Unix offers system calls that let you get more advanced information about files and directories, such as the owner and the attributes of a file.

System files and information: There exist various system calls that let you read and write to special files, such as **/etc/passwd**. Additionally, there are system calls that let you acquire information, such as the current time of a system.

Controlling processes: Linux enables you to create new processes, execute programs, terminate processes and supplies various system calls that can help you do the job.

Threads: Threads enable a single process to perform multiple tasks. Unix offers system calls that let you create, synchronise, etc your threads.

Server processes: Writing code that can create a server process has some unique requirements and certain techniques that use specific system calls.

Interprocess communication: Certain system calls enable processes to communicate with each other using pipes, FIFOs, message queues, semaphores and shared memory.

Network programming: Although network programming from a Unix perspective looks a lot like reading from a regular file, there are certain system calls that are unique to network programming. In particular, such system calls as functions that have to do with resolving hostnames and converting IP addresses are stored as strings to a format that can be used in a Linux system.

Signal processing: Signals provide a way of handling asynchronous events. Almost all applications will need to be able to handle signals.

States of a Linux process

46mm

Back when Unix was first introduced, computers had single CPUs without multiple cores and a very small amount of RAM. However, Unix was a multi-user and multi-tasking operating system. In order to actually be multi-user and multi-tasking, it had to be able to run each individual process sporadically, which means that a process should have multiple states.

There are three categories of processes: user processes, daemon processes and kernel processes. User processes run in user space and usually have no special access rights, whereas kernel processes are executed in kernel space only and can fully access all kernel data structures. Daemon processes are programs that can be found in the user space and run in the background without the need for a terminal. A dedicated user other than root owns them and usually they have more rights than an ordinary user process, but fewer privileges than a kernel process. A web server is a daemon process, the cp utility is a user process and the kworker is a kernel process.

It is important to recognise that a process cannot go directly from state one to state three, or state four, without going into state two first. You will learn more about process states in forthcoming articles. For now, it is sufficient to understand that you cannot control the state of a process; this is the job of the Linux scheduler that runs in the kernel. Put simply, you cannot know when the state of a process will change or when the process is going to go into a running state again.

PROGRAM A CAT UTILITY

Rebuild the concatenation tool from scratch

A C system call is just a normal C function. What makes a system call different from an ordinary C function is the fact that system calls implement functionality that is usually low level and connects the Unix kernel with the rest of the world. Usually, it is a hardware-related service (accessing a hard disk or a network device), process management (creating or deleting a process) or communicating with core kernel services (process scheduling). In other words, system calls enable programs to request services from the operating system that cannot otherwise be accessed by a process.

The programs that use system calls are often called system programs. You can think of system programs as containers of system calls that help you perform tasks that deal with the internals of your Linux system.

As a general principle, you should always check the return values of the C system calls. This is in order to catch any errors made as soon as possible.

myCat.c

```
001 #include <stdio.h>
002 #include <sys/types.h>
003 #include <sys/uio.h>
004 #include <unistd.h>
005 #include <fcntl.h>
006
007 void cat(int f, char *s)
008 {
009     char buf[2048];
010     long n;
011     while ((n=read(f, buf, (long)
012          sizeof(buf))) > 0)
013         if(write(1, buf, n) != n)
014             printf("Error while printing
015                 output!\n");
016
017     if(n<0)
018         printf("Error reading %s!\n", s);
019 }
020
021 int main(int argc, char **argv)
022 {
023     if (argc == 1)
024         printf("You have to give a
025                 filename!\n");
026
027     int f = open(argv[1], O_RDONLY);
028     if (f < 0)
029         printf("Cannot open file!\n");
030     else
031         cat(f, argv[1]);
032
033     return 0;
034 }
```

Build your first tool

The myCat.c code (bottom-left) natively implements the basic functionality of the famous cat utility and needs at least a valid filename as its command line argument. Please note that the line numbers are added for better referencing the code – you should not type them. The use of a separate **cat** function is not mandatory, but it is considered good programming practice.

It is very important to understand that you cannot be sure about the way the file is going to be read by the operating system. It might read the entire file at once, or read it and print it character by character. While doing any kind of programming, but especially systems programming, you should never make any similar assumptions.

The following command will compile the myCat.c C code using the gcc compiler and should generate an executable program named myCat:

```
$ gcc -Wall -o myCat myCat.c
```

Should you wish to change the name of the executable, you should put a different string after the **–o** parameter. The **-Wall** parameter is for turning on all compiler warnings, which is especially useful during development. Try to correct not only the errors but also the warnings, because it will save you time in the long run.

Lines 1-5 are used for including various C header files in your program. Header files are used for loading libraries that do not belong to the core C language, which in itself is pretty minimalistic.

If you want to know which header files need to be included for a given system call, you should check the man page of the system call. System calls can be found in the second section of man pages – this is especially useful to know for system

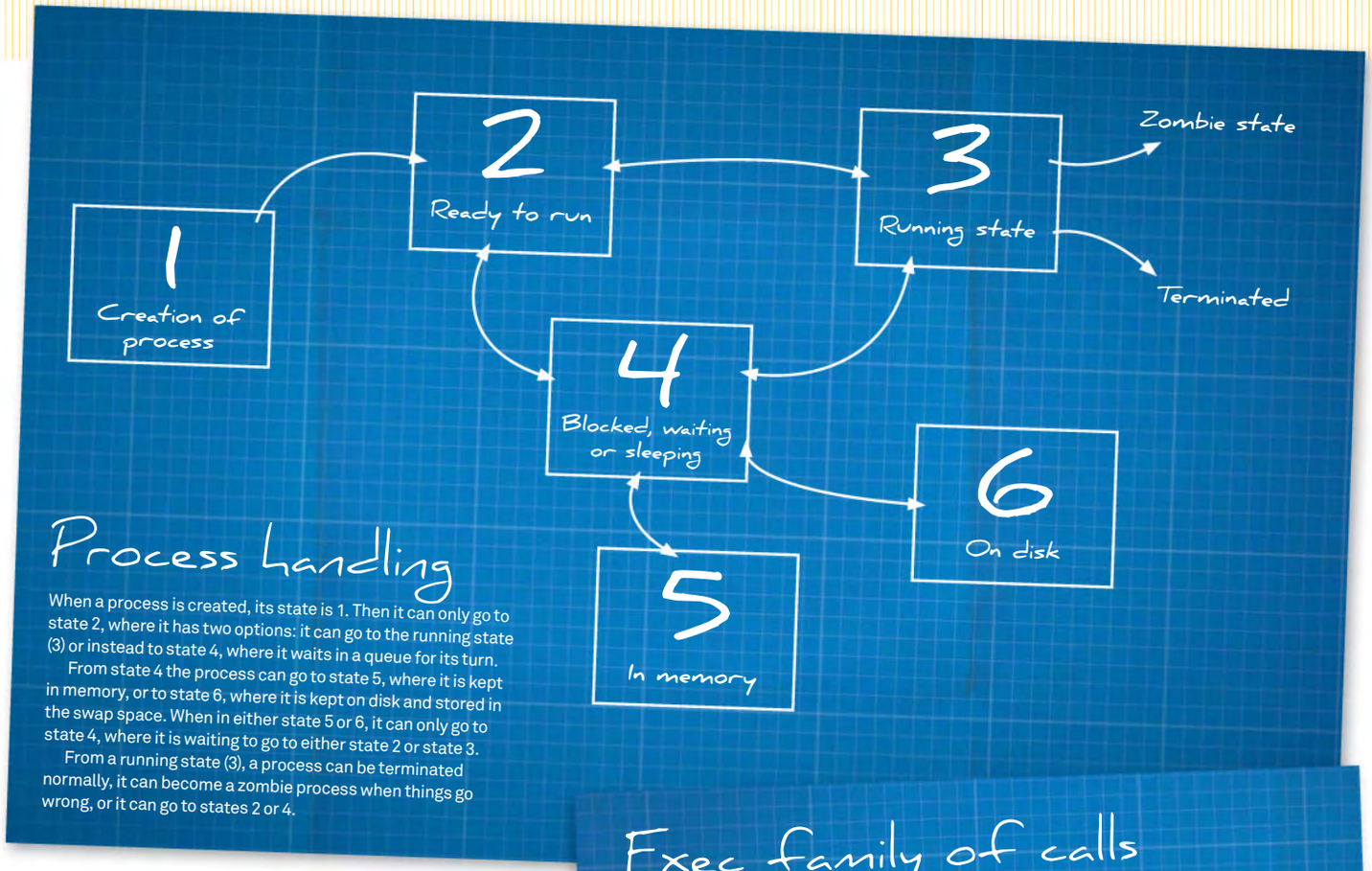
File descriptors

33mm

Although you are going to learn more about file descriptors in the next article, it is good to know some basic things now. A file descriptor is a positive integer value.

Unix supports three special and standard filenames: /dev/stdin, /dev/stdout and /dev/stderr. These can also be accessed using file descriptors 0, 1 and 2, respectively. Alternatively, file descriptor 0 can be accessed using /dev/fd/0 – remember that in Unix, everything is a file.

As you can see in the myCat.c code listing to the left, the **cat()** function writes to file descriptor 1 (the '1' argument in line 12), which is the standard output. In other words, what is written to file descriptor 1 is printed on the screen. This is also analogous to the functionality of the **printf()** function. Similarly, you can read the user input by getting data from file descriptor 0. This is similar to the functionality that is offered by the **scanf()** and **getc()** functions.



Process handling

When a process is created, its state is 1. Then it can only go to state 2, where it has two options: it can go to the running state (3) or instead to state 4, where it waits in a queue for its turn.

From state 4 the process can go to state 5, where it is kept in memory, or to state 6, where it is kept on disk and stored in the swap space. When in either state 5 or 6, it can only go to state 4, where it is waiting to go to either state 2 or state 3.

From a running state (3), a process can be terminated normally, it can become a zombie process when things go wrong, or it can go to states 2 or 4.

Exec family of calls

Every Unix shell uses one of the functions in the exec family to execute commands given by the user. There exist six functions in the family: **execl**, **execlp**, **execle**, **execv**, **execvp** and **execvpe**. The exec functions will return -1 if an error has occurred; otherwise they return nothing. In one of the upcoming articles, you are going to learn more about the similarities and the differences between the six functions.

When a process uses one of the exec system calls to execute a program, the new program completely replaces the process and starts its execution at its **main()** function. However, the process ID remains the same!

The somewhat related **fork(2)** system call that you are going to learn in another article is used for creating a new child process, whereas an exec function starts a new program.

calls that have the same name as some built-in commands. For example, the man page of the **open()** system call can be accessed by executing **man 2 open**; if you just type **man open** then you are going to get the man page of the **open** command that is part of the shell. As a general bit of advice, it is better to use the man page section number. Most of the time, the header files of related system calls are overlapping and should be included only once.

In line 7, a new function named **cat** is implemented. The **cat** function reads from a file using a file descriptor. The file descriptor is passed as a parameter to the **cat** function because it has been opened inside the **main()** function. The second parameter to the function is the filename, which is a string. As you already have the file descriptor, you do not need the filename to read the file. The name of the file is needed in order to have an informative error message, in case an error occurs during the reading of the file. In line 12, you use the **write(2)** system call to print the output on screen using file descriptor 1, which is always standard output – usually your terminal. You also see some error checking here: the **write(2)** function must write its whole input that is stored in the **buf[]** variable. If it fails, then an error message is printed on screen.

In line 19, you see the standard definition of the **main()** C function, where every autonomous C program automatically starts its execution.

In line 24, you see how to open a file for reading using the **open(2)** system call. The **open(2)** system call has two parameters: the name of the file to open and **O_RDONLY**, which denotes that you open the file for reading only. The general principle with the flag values is that you use the flag with the 'smallest' permission that does your job for security reasons. Therefore if you want to open a file for writing only where the **O_WRONLY** flag is enough, you should not open it using the **O_RDWR** flag as that opens it for both reading and writing. When a program ends, all open file descriptors are then automatically closed.

In case of success, the **open(2)** system call returns a positive integer (line 25), which is the file descriptor of the file you opened. In case of error, the **open(2)** system call returns a negative integer. In the next article, you are going to learn more about file I/O and the various flags of the **open(2)** system call. If you want to manually close an open file descriptor – which is good practice – you should use the **close(2)** system call.

The cat program is the perfect example of just how little C code it takes to develop the core functionality of a fundamental Unix utility.

RE-CREATE CORE TOOLS

Build simple versions of ls and wc, and work with process IDs

In this part we will program simple versions of two existing Linux tools to help you understand that, behind the scenes, things are simpler than you might think. We'll also show how to get the process ID as well as the parent ID of a program.

Despite the simplicity, you should always remember that the difficulty in programming systems software lies in the numerous details of a program, for instance error checking, command line processing, error handling and so on. The good thing with such details is that you should be able to write the code once and then use it everywhere!

As you can also see, all programs have comments. Even if a program is small, you should always put the necessary comments in it.

Warning: the C programming language assumes that the programmer always knows what they are doing and does not question them unless there is an error in the code. So, if there is an error, the C compiler will not generate an executable. If there exists any number of warning messages, but no errors, then the compiler will create an executable, which is not always good. As a general principle, you should always try to resolve not only error messages, but warnings as well.

Light version of ls

In this section you are going to program a simple version of the ls program. It will just print the files and sub-directories of the directory that will be given as a command line argument. Type in the code shown to the right, or grab it from FileSilo. As usual, you should compile it using the following command:

```
$ gcc -Wall -o myLS myLS.c
```

The **DIR** type used in the program (line 14) represents a directory stream and is the Unix way of reading a directory. The **dirent** structure (line 15) contains not only the name of the file or directory, but many other things – you will learn more about the **dirent** structure in a forthcoming tutorial. You can also visit **bit.ly/1EO9WV7**.

The parameter of the myLS program is just a pathname. As you may already know, there are two kinds of pathnames: absolute and relative. Absolute pathnames begin with a slash whereas relative pathnames do not. The **opendir()** system call opens the directory and returns a directory stream (line 18). The **readdir()** system call returns a pointer to a **dirent** structure. You can access all its elements of a directory stream with the help of a **while()** loop (line 25). When you reach the end of the directory stream, NULL is returned and the while loop will end.

If you choose to compile and execute myLS, you will find out that each entry is printed on a separate line as a result of the last **printf()** statement.

Challenge yourself

Once you've worked through these utilities and had a go at re-creating another standard Linux utility of your choice, there's another good exercise for you to try: combine multiple utilities into a single program. You might, for example, try making a tool that combines cat with wc, to give you the word count of the newly-concatenated file.

myLS.c

```
001 #include <fcntl.h>
002 #include <stdlib.h>
003 #include <stdio.h>
004 #include <dirent.h>
005
006 int main(int argc, char **argv)
007 {
008     if (argc != 2)
009     {
010         printf("Please provide the correct number of arguments!\n");
011         return -1;
012     }
013
014     DIR *theDirectory;
015     struct dirent *entriesP = NULL;
016
017     // Open the directory
018     if (( theDirectory = opendir(argv[1])) == NULL )
019     {
020         printf("Cannot open directory %s.\n", argv[1]);
021         return -1;
022     }
023
024     // Read all its entries
025     while ( (entriesP = readdir(theDirectory)) != NULL )
026     {
027         printf("%s\n", entriesP->d_name);
028     }
029
030     closedir(theDirectory);
031
032     return 0;
033 }
```

myWC.c

```
001 #include <fcntl.h>
002 #include <stdlib.h>
003 #include <stdio.h>
004
005 int main(int argc, char **argv)
006 {
007     if (argc != 2)
008     {
009         printf("Please provide the correct number of arguments!\n");
010         return -1;
011     }
012
013     int inword = 0;
014     int numberOfWords = 0;
015     // Get the filename of the text file
016     char *file = argv[1];
017     // A character variable
018     char ch = ' ';
019     // Open text file for reading
020     FILE *fp = fopen(file,"r");
```

```
021     if (fp == NULL)
022     {
023         printf("Error reading %s\n", file);
024         return 1;
025     }
026
027     // Start counting words
028     while( (ch = fgetc(fp)) != EOF )
029     {
030         if (ch == ' ' || ch == '\n' || ch == '\t')
031         {
032                 inword = 0;
033         }
034         else if (inword == 0)
035         {
036                 inword = 1;
037                 numberOfWords++;
038         }
039     }
040
041     // Close the file
042     fclose(fp);
043
044     printf("Number of words found %d\n", numberOfWords);
045
046     return 0;
047 }
```

pid.c

```
001 #include <stdlib.h>
002 #include <stdio.h>
003 #include <unistd.h>
004
005 int main(int argc, char **argv)
006 {
007     if (argc != 1)
008     {
009         printf("There is no need for command line arguments!\n");
010         return -1;
011     }
012
013     int pid = getpid();
014     int ppid = getppid();
015
016     if ( pid > 0 )
017     {
018         printf("Your process id is %d\n", pid);
019     }
020     else
021     {
022         printf("There is an error with your process id %d\n", pid);
023     }
024
025     if ( ppid > 0 )
026     {
027         printf("You have been called from a process with id %d!\n", ppid);
028     }
029     else
030     {
031         printf("There was an error while getting parent process id:
032                 d!\n", ppid);
033     }
034
035     return 0;
036 }
```

Error messages and bugs

Usually, your code will have some errors or bugs. A common error is forgetting to include the proper header files in your program. If you try to compile myLS.c without including the dirent.h file, you will get one error message and one warning:

```
yLS.c:13:5: error: unknown type name 'DIR'
     DIR *theDirectory;
 myLS.c:17:5: warning: implicit declaration of function 'opendir'
...
```

If you misspell the **theDirectory** variable, you will get the following error message:

```
myLS.c: In function 'main':
myLS.c:18:11: error: 'theDirecory' undeclared (first use in this function)
     if (( theDirecory = opendir(argv[1])) == NULL )
```

Now, let us go to line 25. If you write **(entriesP == readdir(theDirectory)** instead of **(entriesP = readdir(theDirectory)**, you are going to get a warning at a completely irrelevant place (line 27):

```
myLS.c:27:24: warning: 'entriesP' may be used uninitialized in this
 function [-Wmaybe-uninitialized]
         printf("%s\n", entriesP->d_name);
```

Nevertheless, the gcc compiler will create an executable file because the C code is syntactically correct as far as C is concerned. The problem is that your program will produce no output, as the **entriesP** variable has no value because you used a comparison operator (==) instead of the assignment operator (=). Congratulations – this is your first bug!

87mm

Simple spin of wc

This section will show you how to create a simple version of the wc utility that will only count the words in a file – the myWC program needs just one command line argument, which is the filename that is going to be read.

You should know how to compile your code by now so go ahead. Try to pay attention to any error or warning messages. The code for myWC is bigger than the other two programs because it does more things; the logic of the program is encapsulated in the while loop.

There are many more efficient ways to read a file instead of character by character, but in this case, this is the simplest solution because you are going to do three character comparisons afterwards. Therefore the **fgetc()** function (line 29) is used for reading the whole input one character at a time. The program assumes that when you read a space character, a tab or a newline character (line 31), you are not inside a word. The first time you see a character that is different from space, tab or newline, the program assumes that you are at the beginning of a new word (line 35).

The next systems programming tutorial will build a version of wc that's closer to the /usr/bin/wc utility on your Linux.

Process ID and parent process ID

This program shows that it is not always necessary to write many lines of code in order to get the information you want.

Lines 13 and 14 do the job by calling two system calls that belong to the same family: **getpid()** and **getppid()**. Both system calls require no arguments; each one just returns the process ID that was asked for. The checking in line 7 is not required, but it is good to inform the user that they do not have to give command line arguments for this command. The process ID of the parent process will most probably be the process ID of your current shell.

You might say that there is too much error checking going on here and you might be right. However, the general principle is that the more critical the application you are writing is, the more error checking code it should have.

GO BEYOND C

There are many other programming languages for you to work with, although none are as mature as C

There exist many programming languages that can help you perform system tasks. All programming languages implement their own functions by using C system calls because it is the only interface for talking to the Linux kernel.

The example program that is going to be implemented on all six programming languages is a minimalistic version of the cp command line utility. The presented implementations in C++, Rust, Go, Perl and Python are not the only ones enabled or offered by each programming language. They are presented here in order to give you a quick sense of each programming language.

C and C++

When it comes to systems programming, C++ can be almost identical to C. Their only difference is that in C++, you can organise your code differently because C++ is an object-oriented programming language.

Just to the right, you can see both the C and C++ versions of the cp utility. The C++ version uses an unorthodox approach and performs no error checking, whereas the C code is, more or less, the standard way of writing the cp utility. There exist other techniques that can be used for file copying, but the general idea will be the same. The biggest advantage of the C approach is that if an error occurs during the execution of the program, you will know exactly which system call caused the error. Think about it a little: you will know the precise line of the C code and the exact reason behind the problem! You cannot tell the same from the C++ version of the program because it uses higher-level commands that do many things using just one command.

It's not strange that the C code of the cp command is the longest of all presented code as the functions are primitive.

Best language?

You may ask which is the best programming language for Linux systems programming. The easy answer is C, because Unix is written in C and C is more mature as a programming language than the alternatives.

The main advantage of programming languages such as Perl and Python is the fact that they can easily interact with other things like databases. Additionally, they can implement a lot of functionality using just a few lines of code. However, they hide too many low-level details from programmers and make decisions for them.

Both Rust and Go have the potential to become the next big thing in systems programming, and you should follow their progress. Another interesting programming language that you might want to check, as soon as the necessary tools become available on Linux, is Swift from Apple.

46mm

Kernel development

Since the Linux kernel is written in C, you might wonder how much of this applies to kernel development. The good news is that system call handling and other techniques discussed here will be invaluable, although there are key differences – a **main()** function isn't required in kernel modules, for example, because kernel programming is asynchronous. It's worth dipping into the Foundation's guide: **bit.ly/1K7UHCj**.

C

```c
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

int myCP(char *file1, char *file2)
{
    FILE *fRead, *fWrite;
    fRead = fopen(file1, "r");
    fWrite = fopen(file2, "w");

    if (fRead == NULL) {
        printf("Could not open %s!\n", file1);
        return -1;
    }

    if (fWrite == NULL) {
        printf("Could not create %s!\n", file2);
        return -1;
    }

    size_t l1;
    unsigned char buffer[8192];
    while ( (l1 = fread(buffer, 1, sizeof buffer, fRead)) > 0 ) {
        size_t l2 = fwrite(buffer, 1, l1, fWrite);
        if (l2 != l1)
            printf("There was an error: %zu <> %zu\n", l1, l2);
    }

    fclose(fRead);
    fclose(fWrite);
    return 0;
}

int main(int argc, char **argv)
{
    if (argc != 3) {
        printf("Please provide the correct number of arguments!\n");
        exit(-3);
    }
    return myCP(argv[1], argv[2]);
}
```

C++

```cpp
#include <iostream>
#include <fstream>

int main(int argc, char **argv)
{
     std::ifstream src(argv[1], std::ios::binary);
     std::ofstream dst(argv[2], std::ios::binary);

     dst << src.rdbuf();
}
```

Go

```
package main
import (
    "fmt"
    "os"
    "io/ioutil"
)

func cp(source, destination string) {
    data, err := ioutil.ReadFile(source)
    if err != nil {
        fmt.Println("Error reading the file!")
    }
    err = ioutil.WriteFile(destination, data, 0644)
    if err != nil {
        fmt.Println("Error copying the file!")
    }
}

func main() {
    if len(os.Args) != 3 {
        fmt.Println("You should use exactly")
        fmt.Println ("two command line arguments!")
        os.Exit(-1)
    }
    cp(os.Args[1], os.Args[2])
}
```

Rust

```
use std::env;
use std::fs;

fn main()
{
    let args: Vec<_> = env::args().collect();
    if args.len() == 3
    {
        let input = ::std::env::args().nth(1).unwrap();
        println!("input: {}", input);
        let output = ::std::env::args().nth(2).unwrap();
        println!("output: {}", output);
        fs::copy(input, output);
    }
}
```

Python

```
#!/usr/bin/python
import sys
import shutil

if len(sys.argv) == 3:
    input = sys.argv[1]
    output = sys.argv[2]
    shutil.copy2(input, output)
```

Perl

```
#!/usr/bin/perl -w
use strict;
use File::Copy;
my $input = shift;
my $output = shift;

copy($input, $output) ||
    die "File copying failed: $!";

exit 0;
```

Go

Go is an open source programming language that makes it easy to build reliable and efficient software.

The code to the left shows the Go version of the cp utility, which uses a pretty unconventional approach as it reads the whole file and then copies it all at once – this method might not be so efficient for really large files.

Apart from necessary error checking, the **cp()** function does the job using two Go functions without the programmer dealing with any other issues. You can generate an executable file by running **go build copy.go**.

Rust

Rust is a new systems programming language that tries to avoid unpleasant bugs caused by unsafe code. The compiler is clever enough to produce warnings and errors with useful messages that assist you in avoiding bugs and correcting errors.

The next piece of code shows the Rust version of our utility. The good thing is that it also copies the permission bits of the original file to the destination file. However, Rust code looks unpleasant and it's hard to read; the C code is more beautiful.

At the time of writing, the current version of Rust is 1.2. Nonetheless, Rust is under development, which means your program might need changing when a new version comes.

Python

Python can also be used for systems programming. It might not be the best option to write a highly available production web server, but it can be used for testing and prototyping with great success. You can also write useful command line utilities without the need to write too much code, as you can see from the code showing the Python version of the cp utility.

Perl

Perl is a well-known programming language that can do almost anything including systems programming. It might not be the best language to write a production DNS server that accepts thousands of requests, but it is a very capable programming language for developing useful command-line utilities.

The code at the bottom of the page shows the Perl version of the cp utility. It is small, comprehensive and easy to read.

Comparing languages

Despite the phenomenally different implementation of cp, each programming language uses the same C system calls under the hood to perform similar tasks. In other words, Unix enables you to read from a file using certain methods – certain system calls – that you cannot get away with because the Unix way is strict! So, the Perl implementation of the cp command will have to use C system calls from the same classes of C system calls as the Python implementation. Nevertheless, the actual C implementations of Perl and Python – especially the algorithms – may differ depending on how they deal with error checking, error corrections and so on.

This can be easily revealed with the help of the **strace(1)** command that traces system calls. Other Unix variants have similar utilities to strace for the same purpose. Each line in the strace output shows the name of the system call, its arguments and its return value. As a side effect, strace can also be used for finding out why a program fails to execute.

You should understand by now that usually the main benefit from using a programming language other than C is simplicity. This is because you have to write, debug and maintain less code. Also, you should realise that if you want your program to be fast, you would need to write lower level code using a lower level language.

The next articles in this series will contain more interesting code and perform many exciting tasks related to file I/O, so stay tuned!

Troubleshoot & Repair Linux networks

No network connection on your laptop or problems with your Web hosting? We're here to help

"The Network is the computer," is the famous, prescient quote made by Sun Microsystem's chief scientist and employee number five, John Gage, in 1984. The growth of the web, mobile and cloud computing have borne out that phrase, and a computer without a network connection is just an expensive paperweight.

Fortunately networking is central to Linux, with the Internet, and the Web, having been built on UNIX. Most distros have built-in tools that will tell you what's going on, or at least start you off investigating network problems. Sophisticated tools can be found in your distro's repository and, as nearly all are command-line based, will work as well on your VPS as on a laptop.

We'll take you through the basics of the GNU/Linux networking stack, and what can go wrong with it (and the rest of the Internet). We'll look at tools and config files to help you and finish with help for netcat, dig, traceroute and Wireshark.

Network essentials

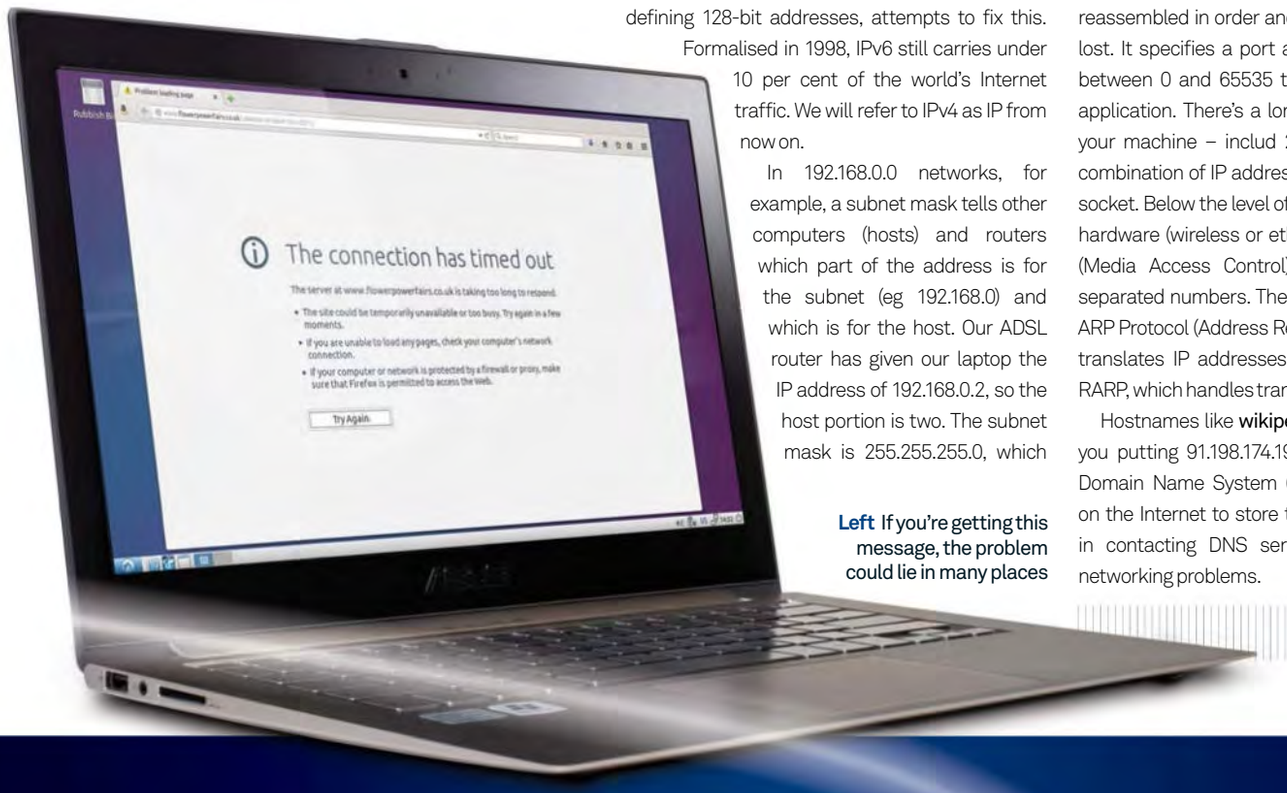
The first step to troubleshooting your Linux network is to fully understand how it works

Where is the network down? Don't neglect hardware problems – after basic checks it's worth looking for pulled cables or fault lights on your Wi-Fi router. Some problems are easy to check, while some are more likely than others – let this guide you in the order you tackle your search.

You don't need to pass an RHCE (Red Hat Certified Engineer) or LPI (Linux Professional Institute) exam, you just need an appreciation of TCP/IP networking. Feel free to skip through this page lightly, and refer back after reading the more practical parts of the article.

TCP

TCP/IP – Transmission Control Protocol/Internet Protocol – is a set of rules for computers to communicate. TCP sits on top of IP, confirming each data packet sent – lots of overhead compared to UDP (User Datagram Protocol) where no checks are made. It does mean useful information is there for tools to diagnose problems.



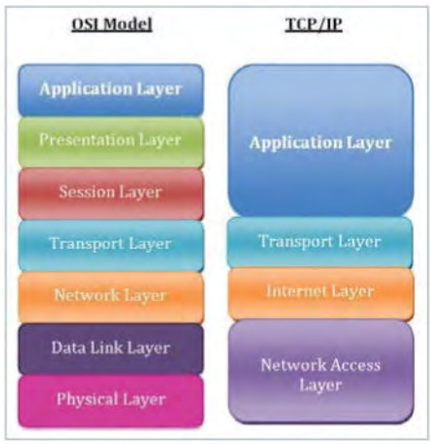
Left If you're getting this message, the problem could lie in many places

IP, the Internet layer, defines the datagram – the basic unit of transmission in the Internet, consisting of a header and a block of data. The header contains the information needed to deliver it – routing from the originating equipment to the destination – in five or six 32-bit words.

The header contains the destination address for the data. If it's not on the local network, it will be passed to a gateway (or IP router) and continue until it reaches its destination, its journey being determined by routing protocols. The address in IP version 4 (IPv4) is a dotted quad, a 32-bit binary number normally expressed in the form n.n.n.n, where n is anywhere between zero and 255. Certain numbers are reserved, such as 127.0.0.1 for local host, a way for any computer to refer to "myself", and private addresses used for local networks, such as 192.168.n.n.

When even your toaster wants to connect to the Internet, the 4.3 billion addresses provided by IPv4 aren't enough. IPv6 (version 5 never got going) defining 128-bit addresses, attempts to fix this. Formalised in 1998, IPv6 still carries under 10 per cent of the world's Internet traffic. We will refer to IPv4 as IP from now on.

In 192.168.0.0 networks, for example, a subnet mask tells other computers (hosts) and routers which part of the address is for the subnet (eg 192.168.0) and which is for the host. Our ADSL router has given our laptop the IP address of 192.168.0.2, so the host portion is two. The subnet mask is 255.255.255.0, which tells routing devices what parts of the IP address to treat as what.



Above Know your onions: compare the OSI seven-layer model with the four-layer TCP/IP model

Credit: CC-SA-3.0 - http://en.wikipedia.org/wiki/File:Application_Layer.png

TCP establishes a connection between a destination and source, ensuring packets are reassembled in order and re-sending any that get lost. It specifies a port at each end – numbered between 0 and 65535 to indicate the service or application. There's a long list in /etc/services on your machine – includ 25 for sending mail. The combination of IP address and port is known as a socket. Below the level of IP, your physical network hardware (wireless or ethernet card) uses a MAC (Media Access Control) address – six colon-separated numbers. The protocols for this are the ARP Protocol (Address Resolution Protocol), which translates IP addresses to MAC addresses and RARP, which handles translation the other way.

Hostnames like **wikipedia.org** are used to save you putting 91.198.174.192 into the browser. The Domain Name System (DNS) uses DNS servers on the Internet to store these names, and issues in contacting DNS servers account for many networking problems.

Diagnosing issues

Finding the root problem can be tricky, but there are a number of places you can look first

What's not working – connecting to one website or all of them? If it's just one then it may still be a problem at your end, but if it's everything, let's find out where the problem lies.

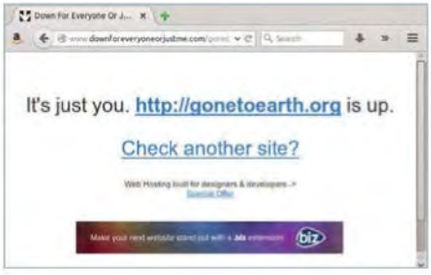
First your network connection – most desktop distros ship with NetworkManager to manage connections. From the command line, typing **nm-tool** will report what it knows of your network – look for 'State: connected'. If you don't have nm-tool, use **ifconfig** to see which interfaces are recognised and **ethtool** for connection status information, or use **iwconfig** for wireless connections.

While ethtool will show you're physically connected to the network (Link detected: yes) and iwconfig that you are connected to a wireless router, ifconfig will give you your IP address and netmask, telling you that this much of your networking is successfully configured.

Running **route** will show the routing table, which includes the default gateway to the rest of the Internet. If there's no default gateway shown for addresses outside the local subnet, you will need to fix this. Route can be used to add routes but you need to address the cause of the problem.

Your servers will have fixed IP addresses, which can be edited to correct gateway and other network details. Laptops tend to be configured automatically by a DHCP (Dynamic Host Configuration Protocol) daemon, often running on an ADSL router, where settings can be changed for the problem machine if necessary. Having corrected settings, a network restart:

```
sudo service networking restart
```

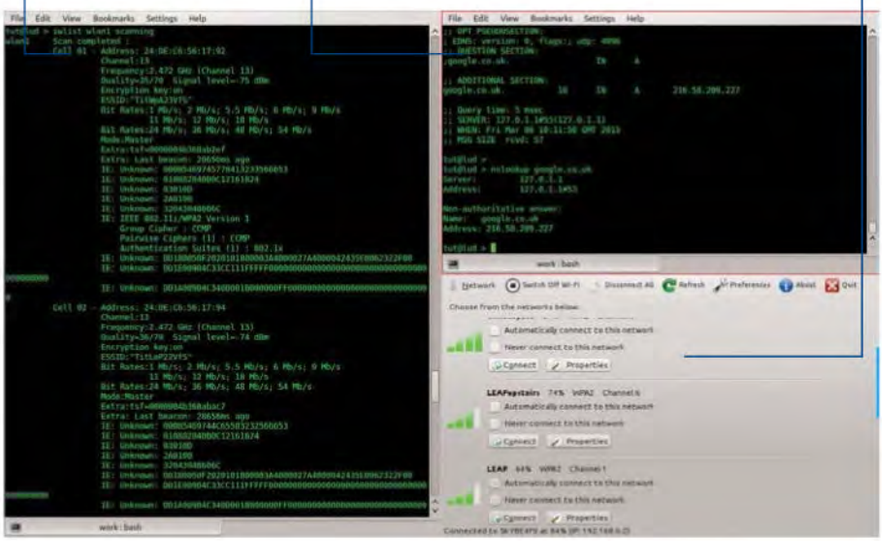


Above **www.downforeveryoneorjustme.com** is a very handy diagnostic tool, simple as it sounds

Can't connect to Wi-Fi? The iwlist tool shows you everything your wireless network interface can see

Nslookup gives you the domain's IP address, and where it looked for it. Simple, but effective

A text-based interface and a scriptable version are available for Wicd, but the GUI frontend is fine



...will pick up the revised settings on Debian-based PCs – leave out the gerund (the -ing) for Red Hat boxes. Run route again to check for the appearance of the default gateway.

Ping uses another part of the TCP/IP protocol stack, ICMP, to send an ECHO_REQUEST datagram, and the ICMP ECHO_RESPONSE produced by the host or gateway pinged is used to calculate a time for the trip.

Ping tells you if a machine is up, what latency there is in the network and how many packets are lost, all indicative of something unless the server has been set to drop ICMP requests by an overzealous sysadmin, something of negligible security use in most cases.

Use ping to check that you have a route to hosts on the Internet. Start by pinging your gateway:

```
ping 192.168.0.1
```

...then ping a reliable host like 8.8.8.8, one of Google's public-facing DNS servers (the other is 8.8.4.4). We've been using IP addresses and **-n** switches to avoid DNS problems distracting us from other network faults, but now's the time to check DNS functionality. **Nslookup**, less sophisticated than **dig** (part of dig's output can be seen above), but is fine for checking that a domain name resolves to an IP address. If you don't get an answer, have a look in /etc/resolv.conf.

If you've ruled DNS out, try some of the tools overleaf – traceroute to see if you can route all the way there, telnet and friends to see if a particular port is open, dig for more DNS and Wireshark for investigating unresponsive or slow services.

If it is your webserver that's the problem, then **ssh** in and run:

```
netstat -lnp | grep -i apache
```

...(replacing apache with nginx, httpd or whatever is appropriate) to see if your web server is listening to all addresses on port 80. You could **grep** 80 if that's the only port which you're concerned with, but check what else Apache is up and listening on.

Configuration files

Diving into the config files with your favourite text editor is a great way to quickly solve problems

Everything is a file, even connected devices – that's the Linux way. In the Eighties many Unix systems kept binary configurations, but inspired by the Plan9 operating system, Linux put most configuration information in text files. Knowing where they are and what to do with them means your text editor also becomes a powerful tool in checking, fixing and maintaining your Network.

This starts at the hardware level – physical interfaces are found under /dev, and /proc exposes the configuration of installed PCI buses and devices to be read by **lshw** when you call:

```
lshw -C network
```

...to check the logical name entry to use with tools like ethtool and ifconfig.

It's not always simple though. When swapping between Red Hat and Debian/Ubuntu based machines, the ethernet interface on our Ubuntu machine was configured in the file /etc/network/interfaces, while the Fedora 20 laptop's NIC was /etc/sysconfig/network-scripts/ifcfg-em1, sharing a directory with ifcfg-*** files for every wireless hub to which we had ever connected it.

Linux's everything-is-a-file approach also means that if you have issues with hardware, they can often be solved with a text editor. For example, if the kernel isn't loading the module for your NIC then /etc/modules, or a similarly named file on your distro, is the place to add not just modules to load but also alisases to the device's name, if that's what is causing the error:

```
alias eth0 b44
```

DNS again

DNS is accessed by the resolver routines – read the config file /etc/resolv.conf to know where to search. Look at the file on your laptop and you may see something like:

```
As an example your cat /etc/resolv
conf may
 # Dynamic resolv.conf(5) file for glibc
resolver(3) generated by resolvconf(8)
# DO NOT EDIT THIS FILE BY HAND -- YOUR
CHANGES WILL BE OVERWRITTEN
nameserver 127.0.1.1
search Home
```

The 127.0.1.1 (rather than 127.0.0.1) is a pointer to a PC running dnsmasq that is a lightweight forwarding DNS server under the control of NetworkManager. In distros without this, dhclient will grab the address of the DNS server from the DHCP server.

It is best to use /etc/resolvconf/resolv.conf.d/base to place an entry like the following:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

...for automatically writing to /etc/resolv.conf. Then running **resolvconf -u** (as root or with sudo) will update resolvconf.

A closer look at /etc/resolv.conf shows it to be a symlink to /run/resolvconf/resolv.conf, which is where dnsmasq writes it. To temporarily remove dnsmasq, try commenting out its entry in /etc/NetworkManager/NetworkManager.conf.

DNS servers are queried in the order they appear in your /etc/resolv.conf file – put the one you want to try out first and/or comment out the remainder by placing a # at the beginning of its line so that the resolvconf ignores it.

Opennicproject.org and **http://freedns.zone** offer DNS with no redirects and no logging, which is essential if you live in a place where what you do online is monitored or restricted.

Rounding off config files by returning to IPv6, it can be removed systemwide by editing /etc/modprobe.d/aliases to add:

```
alias net-pf-10 ipv6 off
alias net-pf-10 off
```

```
alias ipv6 off
```

...and rebooting. If you rule it out as a problem, remember to put it back again:

```
alias net-pf-10 ipv6
```

Where Am I?

If you are familiar with **whoami**, which tells you which user you're currently logged in as – handy if you su or ssh a lot and risk losing track – you may expect **whereami** to tell you the name of the machine you're logged into. Not so; to do that you type **hostname**, which reads /etc/hostname.

Whereami is a set of useful scripts for detecting which network you've got your laptop plugged into and configuring it accordingly. Particularly handy for those who run lightweight window managers and distros without all the bells and whistles to quickly click on a choice of available Wi-Fi networks, it also lets you tweak known networks with scripts as well as adapt to new connections with minimal intervention.

Running this may help you avoid some connection hassles in the first place, and it's more flexible than wicd.

Is it plugged in?

You may not believe it, but really, these things do happen. It's not the first thing to check – ifconfig and ping will both show that you have a working ethernet connection, or that the Wi-Fi router is up. However, if tests show no connection, that's when you look for loose cables (is the NIC showing a green light?), unplugged routers and any other physical causes.

Don't forget that many laptops have little buttons to switch off the Wi-Fi card (or F-key shortcuts) that can be accidentally pressed. However:

```
rfkill list all
```

...will (usually) show whether or not this is the case. If it's hard-blocked then hit the switch; if soft-blocked then **rfkill unblock all** will usually get your connection up again.

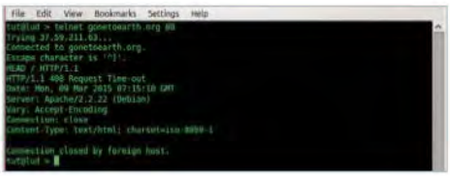
As we said, these things happen, so if it happens to you then just smile because at least it wasn't something more serious that can't be fixed in an instant.

Fix network problems

Using these four apps will help you pin down and fix a number of networking issues

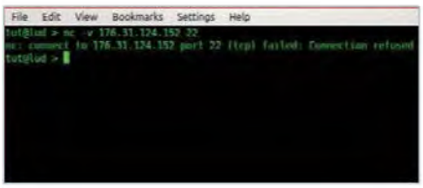
(Telnet to)Netcat

Netcat does everything that the humble telnet does plus much more, but you may find yourself on a box without netcat, so we'll start with an example from old-school telnet.



01 Humble telnet

If you started using computers after the Nineties, when telnet was replaced by SSH in a suddenly far less secure world, you may have dismissed it as a relic from the past. But telnet lives on as a useful diagnostic tool available from any distro, connecting to specific ports to see what's open and working.

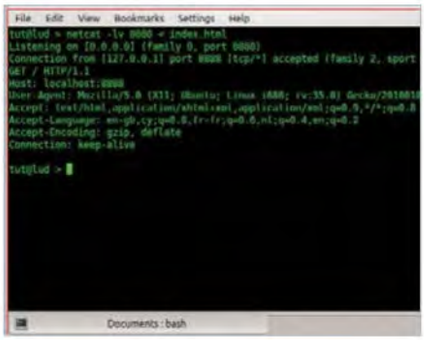


02 Enter netcat

If you can install netcat (nc) then you won't fall back on telnet much, as it combines the simple testing abilities of telnet with abilities to do almost anything with TCP, UDP or Unix-domain sockets: open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports and port scan.

03 Port scan

While it's not good manners to check every port on someone's machine to see what's left open (a portscan), it's useful on your own machines both for security ('that shouldn't be open') and diagnostics ('that should have been open and listening'). Try running something like **nc -vnzu 192.168.0.1 1-65535** to do this.

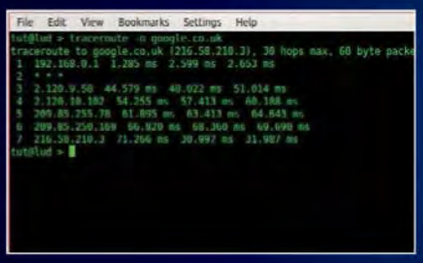


04 Pass the port

One useful nc trick is to quickly set up an impromptu server listening on a particular port, to check there is nothing impeding a connection on that port between you and the server. In the image above, we set nc as a one-off web server and read info on the host that connects to it.

Traceroute

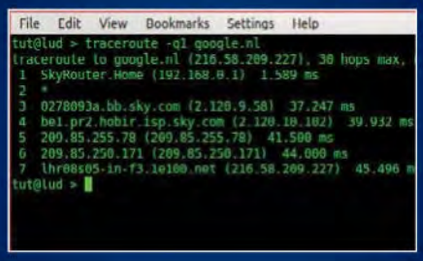
You might not think about how the Internet works while you're using it, but traceroute lifts the lid on where your packets are travelling – showing the time packets take to reach each gateway machine between your machine and the server.



01 Follow the hops

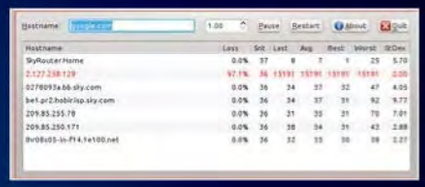
Traceroute tests each hop between you and the destination host. Although not always conclusive, output shows where problems may be occurring. While the screenshot shows the default number of hops and packet size, you can adjust that with:

```
traceroute -m 255 wikipedia.org 70
```



02 Journey times

Those times displayed in ms are the round trip times to each host for three packets sent. Adjust the number of packets with **–q** – for example, **-q1** sends just a single packet. A longer time from the UK could be a channel hop.



03 Mtr

A set of asterixes is an unreachable host but mtr provides a continuous traceroute to help to detect intermittent problems. You may only be able to fix problems found in your own networks, but knowing where the problem lies could help to generate a route around fix.

04 Blocked ICMP

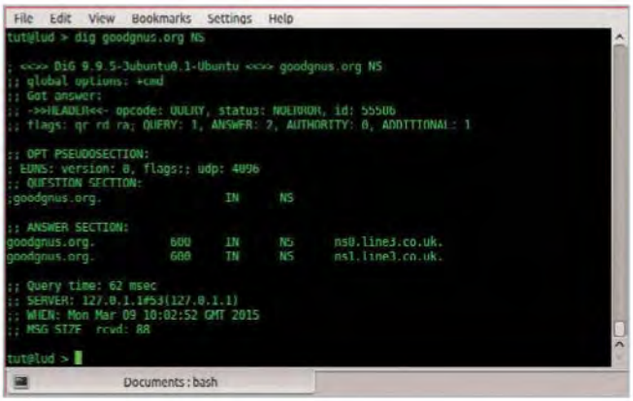
As we mentioned with ping, some systems administrators block ICMP, so standard traceroute won't work. Tcptraceroute provides a traceroute through TCP instead of ICMP.

Dig

We've used the **-n** option a lot in this tutorial, as DNS issues can easily cloud other problems. Once you've cleared up suspected DNS problems on your machine with the resolver, it's time to reach out through the hierarchical world of DNS servers to check everything is as it should be. Nslookup and host perform simple searches, but dig is the most flexible tool available.

01 Address search

Nslookup may be sufficient for resolving an address or checking that you can, but for useful information about DNS servers and their recursive connections across the Internet, fire up dig, whose flexibility means that it repays a little time spent getting to know some of its options.

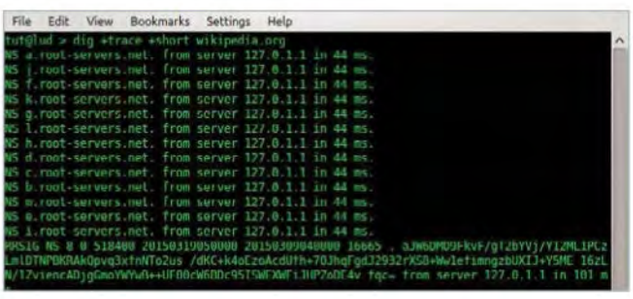


02 Names are served

By default, dig returns A records, but it can be used to check other record types such as MX (mail servers). In the screenshot above we have used NS to find the nameservers for a named domain.

03 Hierarchical

DNS is hierarchical, with the TLD (top level domain), such as .com or .org.uk queried first, then the name part. With searches taking place recursively there is plenty of room for errors – or even, if you're unlucky, malicious attacks. "Dig +trace" shows you the successive hierarchical steps taken by your query.

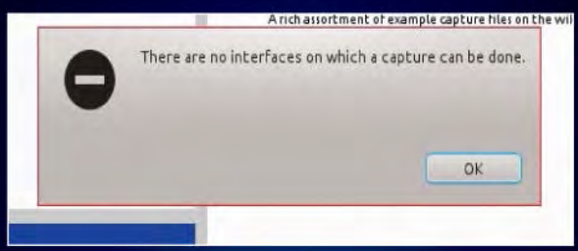


04 +short option

Hierarchical searches output a lot of information that you probably don't need – even from a standard DNS lookup you may only want the IP address. The **+short** option gives you just such an abbreviated output, which is also very useful in scripting searches.

Wireshark

Like tcpdump, Wireshark can dump packets from the network, but it also performs interactive analysis – slightly over the top for minor networking problems, but handy for locating bottlenecks in the system. In most distros Wireshark will be the GUI (Gtk) version, with the console version packaged as tshark. Try them both so you can adapt to whichever is best when trouble strikes.

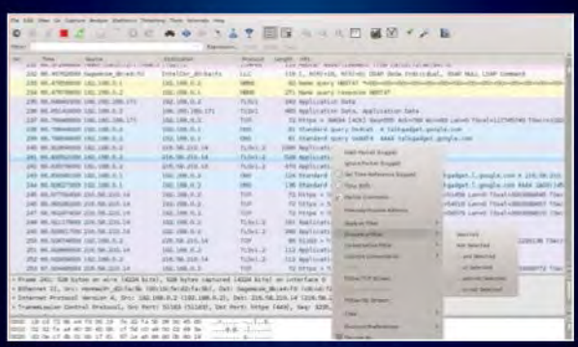


01 Powertool

Despite the baffling number of options available to you, starting is a simple matter of selecting interfaces from the Capture menu. To get Wireshark to see your interfaces and avoid running it as root user, see Capture Setup/Capture Privileges over at **wiki.wireshark.org**.

02 Portable troubleshooting

As Wireshark is useful for detecting many problems with packet loss or latency, and won't be installed everywhere you go, you can avoid the dance around superuser permissions by carrying it around on a USB live distro.



03 Filter cut

Looking at the raw data is overwhelming and even the choice of filters is large, but you can right-click a suspicious entry and use that as the basis for a filter, or do the same from the filter hierarchy. Simplest case, you're looking for a particular protocol – say DNS, or perhaps something encrypted via TLS – so you just put that in the filter toolbar.

04 Command line shark

On your VPS, or other non-GUI box, tshark is functionally equivalent to Wireshark. It's worth installing after Wireshark and then getting familiar with, so you are prepared if you ever need it.

Turn an old PC into a NAS box

Repurpose old hardware with NAS4Free to use as a NAS server for backups and more

Resources

Spare PC with at least 512MB of RAM

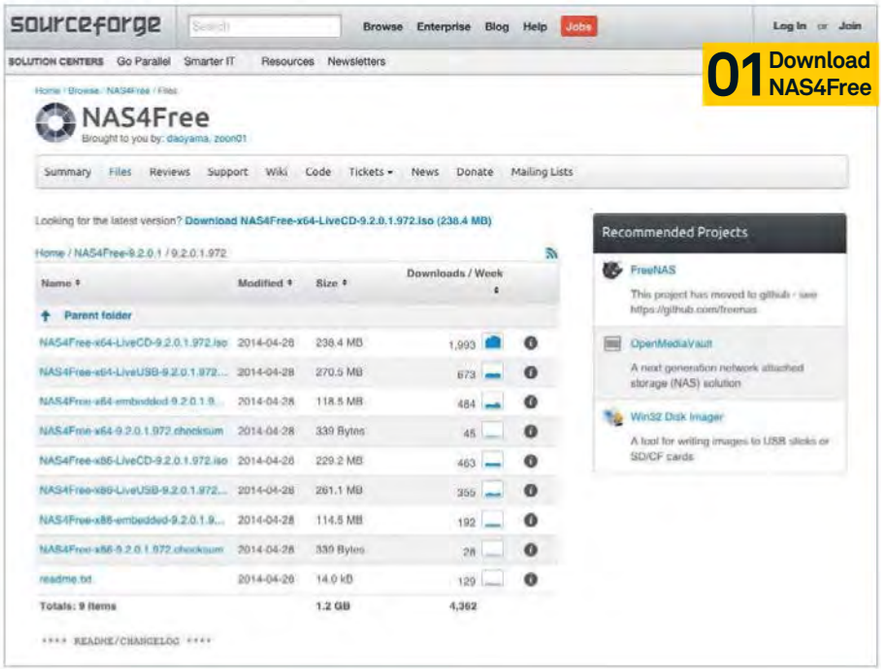
FreeNAS
sourceforge.net/projects/nas4free/files

Home network

The fast pace of technological progress is great, but does mean hardware soon becomes redundant. This begs the question: what do you do with an old PC gathering dust in the attic? One option is to turn it into a network-attached server for storing files, media and backups.

For this purpose there are several specialist distros to choose from, including FreeNAS and OpenMedia Vault. However, to encompass older hardware, we'll be using NAS4Free – a legacy version of FreeNAS – since it has lower system requirements. Officially, it only requires 512MB of RAM to work, but you may be able to get away with as little of 256MB for the Full version.

We'll go through installing NAS4Free on your PC and how to access and configure it remotely from a client PC via its web-based GUI. You can schedule remote backups of some folders with rsync and cron (or Windows Backup or OS X Time Machine). We also cover media streaming and torrents – you could even set up ownCloud hosting. Dust off that old PC get it working again!

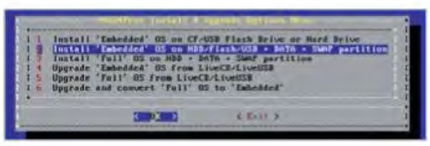


01 Download NAS4Free

You can find the latest NAS4Free files at SourceForge. Choose either a Live CD ISO or Live USB IMG file, depending on whether you want to boot it from CD or USB. Also, select the correct version for your PC: x64 (64-bit) or x86 (32-bit).

02 Boot it up

After setting the BIOS on your old PC so it'll boot first from CD (or the USB stick), insert your live disc/stick and boot it up. NAS4Free will go through the boot process, which may take a while to complete.



03 Choose install method

Eventually you'll get to a Console Menu. Enter 9 to install from your live CD/USB. In the next menu, choose option 2 to install it on the PC's hard disk (or 1 if you want to run the OS from a USB flash drive).

04 Install to disk

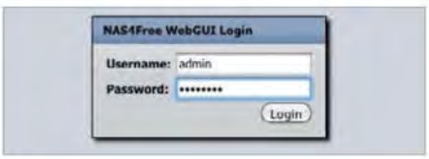
Hit OK on the next menu, choose the installation media and destination media, then say No to a swap partition (unless you have very little RAM). NAS4Free will then be installed on the chosen disk. Note the DATA partition parameters.

05 Configure LAN interface

Now remove the live CD/USB and reboot the computer. After the bootup process, you'll end up back at the same Console Menu. This time, enter 1 to select your Ethernet interface (probably from just one option).

06 Configure IP address

Back at the Console Menu, enter 2 to configure the network IP address. Say No to DHCP and enter a static IP. Press Enter to accept the default subnet mask. Use your router's IP address as the default gateway and enter your favoured DNS.

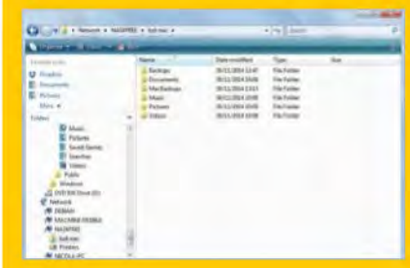


07 Access web GUI

With the basic setup done, you can now access your NAS4Free server from another PC. Just enter its IP address in a web browser and you'll see the NAS4Free web GUI. The default username is 'admin', with password 'nas4free'.

> There are several specialist distros to choose from, including FreeNAS and OpenMedia Vault

Back up OS X and Windows



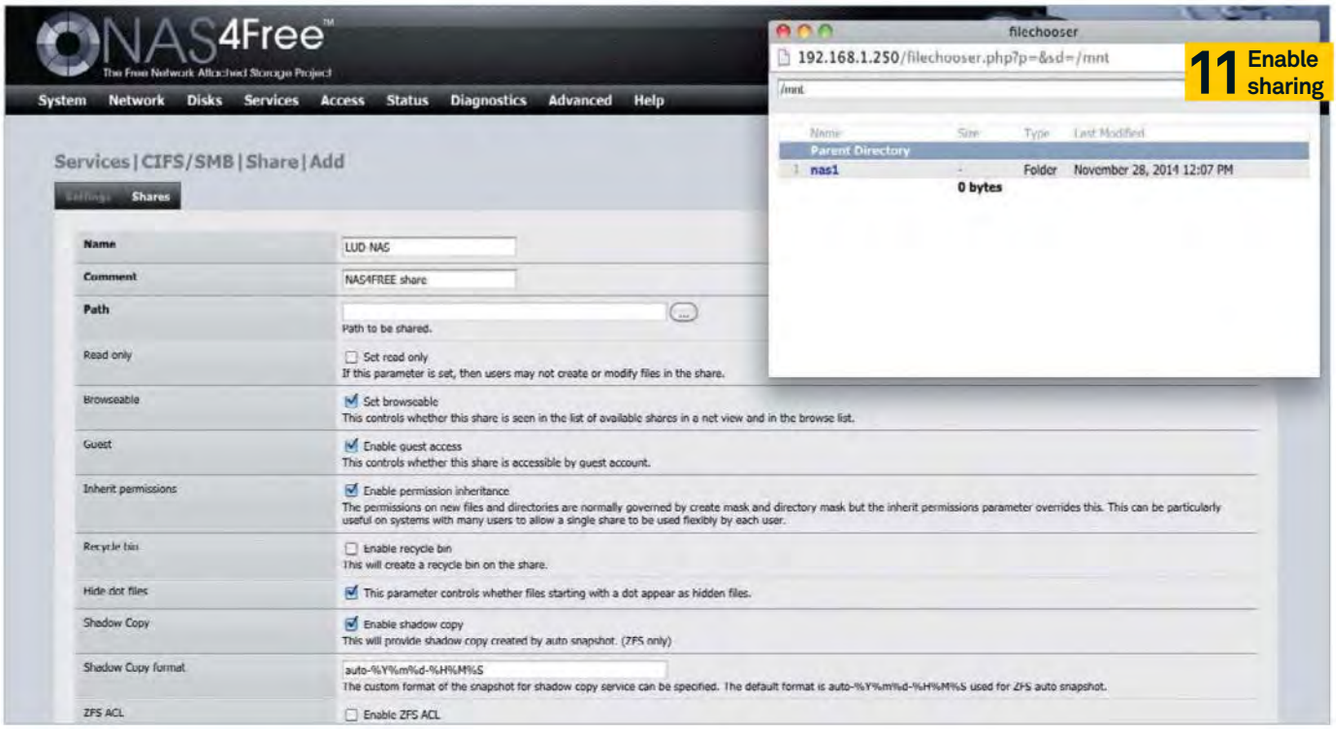
Easily back up a Windows PC

You can access your NAS4Free CIFS/SMB share on a Windows PC by typing \\[your NAS4Free IP address] in the Explorer. While you could back up using rsync, it's easier to use the Windows Backup feature (on Windows 7 Professional or later). Go to Backup & Restore>Set Up Backup, then hit the 'Save on a Network' button. Browse to your NAS4Free shared folder, then click Next, choose backup settings and set the schedule for them.

Back up your Mac via AFP

Again, you could use rsync, but to use Time Machine just share your NAS4Free drive via AFP (Apple Filing Protocol). In the web GUI, go to Services>Users & Groups and click Groups. Click '+', fill in the fields, then Add and Apply Changes. Click Users, then '+' and fill in the fields, assigning the Primary Group as your newly created one. Go to Services>AFP and click Shares. Click '+', add a name and comment, hit the Path '...' button and choose your drive's mount point (and optional subfolder). Enable automatic disk discovery and choose Time Machine. Click Add, then Apply Changes. In Settings, click Enable, tick both authentication options, then Save and Restart. Now, from your Mac's Finder, hit Go>Connect to Server and enter afp://[NAS4Free IP] to connect as a guest. In Time Machine's Preferences, hit Select Disk and you'll see your shared NAS4Free folder.

Enhance your system



Let your NAS box handle all of your torrent downloads with BitTorrent client

Another neat feature of NAS4Free is its built-in Transmission BitTorrent client. From the web GUI, go to Services>BitTorrent and click Enable. Add the download and watch directories, alter any other settings you want, then hit Save and Restart. Now, whenever you add a torrent to the watch folder (from any connected PC), your NAS4Free server will start downloading it. Click the URL at the bottom of the Services>BitTorrent screen to check its progress. Note: you may need to get your router to forward the port used.

08 General settings

For extra security, you can change the username and password via System>General – click the Password tab to change it. The General menu also enables you to alter settings such as DNS and time zone.

09 Add disk

Go to Disks>Management and click the '+' on the right. Choose your hard disk from the drop-down, then the file system for a pre-formatted disk – if it's not, you can format it via Disks>Format. Click Add at the bottom, then Apply Changes.

10 Add mount point

You need to add a mount point for the disk. Go to Disks>Mount Point and click '+'. Choose your disk from the drop-down, keep UFS file system enter partition number 1 and then a mount point name. Click Add, then Apply Changes.

11 Enable sharing

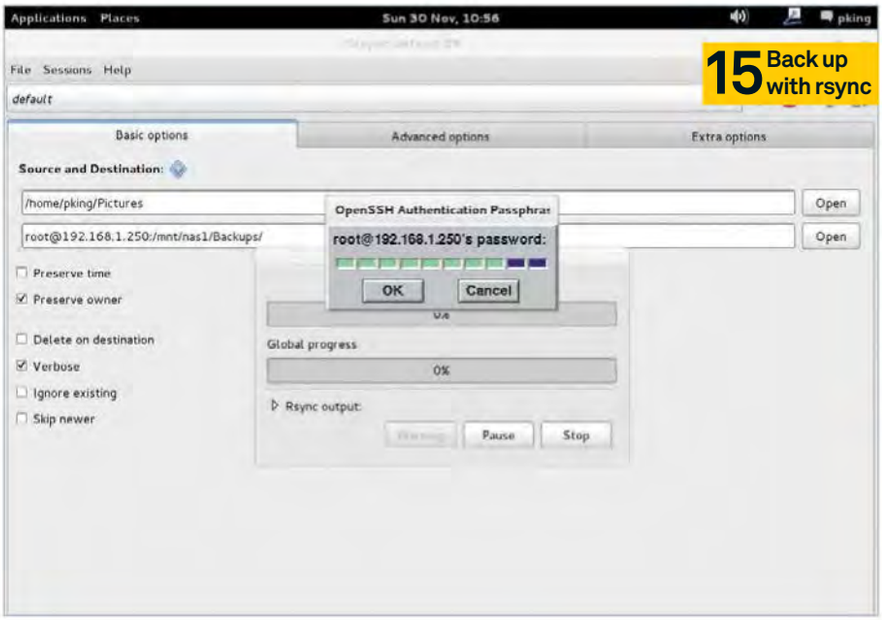
Go to Services>CIFS/SMB and click Enable. Click the Shares tab, then '+' and enter a name and comment. Click '...' for Path and choose your mount point name from the pop-up. Click Add, then Apply Changes. Click the Settings tab, then Save and Restart.

12 Remote access

You can now access the shared folder from the file browser of another PC – Browse Network>Windows Network>WORKGROUP>NAS 4FREE>shared folder. Create a Backups subfolder in it, to separate them from shared files and media.

13 Set up rsync

On the web GUI, go to Services>Rsync. Click the Modules tab, then enter a name and comment. Hit the Path '...' button, select your mount point and Backups subfolder. Click OK, Add, Apply Changes. Click Settings tab, Enable, then Save and Restart.



For extra security, you can change the username and password

14 Start up SSH

We'll want to use rsync with SSH to back up files securely from the client computer to our NAS4Free server. In the web GUI, go to Services>SSH and click Enable. Tick the 'Permit root login' option. Then click Save and Restart.

15 Back up with rsync

Now let's try a manual backup from the client PC. While you can run rsync from the command line, we're using Grsync – a GUI front-end – for ease of use, particularly when choosing options. Choose the folder to back up, then enter the destination: root@[NAS4Free IP]:/mnt/[mount point]/Backups. Click the gears icon and a dialog will then prompt you for a passphrase: enter your NAS4Free password (default is 'nas4free'). The backup will then proceed. This is fine for manual backups, but for automated ones we'll need to set up SSH password-less, key authentication.

16 SSH login

Setting up SSH key authentication (see bit.ly/1zGfaug) is done from the command line. First, open a terminal and enter:

```
ssh -l root [NAS4Free IP]
```

Type 'yes', then enter the password to log in to your NAS4Free server.

17 Generate SSH key

Now we can generate a SSH key pair, just by entering:

```
ssh-keygen
```

Press Enter to accept the default file location, then Enter to set an empty passphrase and Enter again to confirm it. Your SSH key pair will then be generated.

Stream music, videos and photos using UPnP streaming

You can also turn your NAS backup box into a UPnP media server. Make a folder for your UPnP server on the shared disk, via a client PC's file browser or SSH, and subfolders for Music, Photos, Videos (plus Temporary if using transcoding). Go to Services>DLNA/UPnP in the Free4NAS web GUI, click Enable and choose your new folder as the database directory. For the media library, click '...' and browse to one of your subfolders, click OK, then Add. Repeat for the other subfolders. Choose a profile for your UPnP device (eg Sony PlayStation 3). Enable transcoding if needed and select the Temporary directory.

18 Rename public key

Rename your public key with:

```
mv ~/.ssh/id_rsa.pub ~/.ssh/authorized_
    keys
```

Then log out with:

```
exit
```

Copy the private key to your client PC with:

```
scp -p root@[NAS4Free IP]:~/.ssh/id_rsa
    ~/.ssh
```

Enter the password, then SSH in (step 16) and you won't be asked for a password.

19 Automate backups

You can now automate backups with cron. In the terminal, enter:

```
crontab -e
```

Copy and paste your rsync command into a new line at the bottom, preceded by the time and date fields – it's mins, hour, then * * * for a daily backup at that time, so for a 2pm daily backup you'd use:

```
00 14 * * *
```

Debug your own Linux software like a pro

GNU/Linux software-related issues can be solved or circumvented by using a debugger's perspective

Resources

GDB Manual
bit.ly/1GbYIDi

Glibc Heap Consitency-Checking
bit.ly/1Hi5gSC

Why Programs Fail
bit.ly/1KS8j99

The Art Of Debugging
bit.ly/1Bd1jBP

From time to time, a certain piece of software fails. Whether it's due to a defect that lies in its code or within a buggy external library the software depends on, the result is somehow the same: the program usually crashes. In this scenario, an ICT expert is meant to solve the issue or to find a way to circumvent it. However, more often than not, the ICT expert will not have any previous knowledge of the software, so a comprehensive and thorough analysis without the right tools and techniques will prove an insurmountable task. To make matters worse, sometimes the ICT expert will be dealing with proprietary software for which sources will not even be available (bear in mind that there are non-open source programs running on GNU/Linux boxes, too). So a debugger's point of view can save the day. How? This tutorial will examine two real cases where you, playing the role of the ICT expert, will dissect, analyse and eventually fix or bypass the problem by means of thinking like a debugger, making use of the GNU Debugger (GDB) along with the objump utility. The process can work even in those cases where you are not acquainted with the defective software at all.

Bypassing a segmentation fault

Our first case illustrates that, even when we do have the sources but we still do not have a good understanding of the code, it may be feasible to bypass – to an extent – the code implementation's flaws that would otherwise lead us to a crash. For this first real case example, you will face a bug affecting LibreOffice 3.5.4.2: the software crashes with a segmentation fault error message when opening a certain Microsoft Excel XML spreadsheet. You can get a bit more of information about the crash by running **dmesg**:

```
# dmesg|grep soffice
[936028.103160] soffice.bin[3495]: segfault at
200030000 ip 0000000200030000 sp 00007fffff42baa8
error 14 in libunoxmllo.so[7f4e6bfb9000+a2000]
```

... but that's about it. You need to understand what is going on behind the scenes, and the only effective way to do so is by using a debugger's approach. A memory dump tells you something, but the debug symbols can tell you much more. Because this issue happens to a Debian GNU/Linux box, you can install the LibreOffice debug symbols this way:

```
# apt-get install libreoffice-dbg
```

Once you have the debug symbols, you run the program

@Breakpoint at line 1075 to bypass the segfault

```
(gdb) info b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00007fffd60141f4 in ScFormulaCell::Compile(String const&
, bool, formula::FormulaGrammar::Grammar) at /home/rene/Debian/Pakete/LibreOffice/libreoff
ice-3.5.4+dfsg2/sc/source/core/data/cell.cxx:1075
        breakpoint already hit 206 times
        set $check = pCodeOld->nRPN
        printf "Check is %d\n", $check
        if $check>0
          printf "Patching pCodeOld to avoid the crash ..."
          set $hits++
          set var pCodeOld=0x0
        end
        c
(gdb) print $hits
$4 = 2
(gdb)
```

@LibreOffice's segfault and call-stack

```
Program received signal SIGSEGV, Segmentation fault.
0x0000000200030000 in ?? ()
(gdb) bt
#0  0x0000000200030000 in ?? ()
#1  0x00007fffd6014225 in ScFormulaCell::Compile (this=0x7fffe005d770, rFormula=...,
    bNoListening=false, eGrammar=formula::FormulaGrammar::GRAM_NATIVE)
    at /home/rene/Debian/Pakete/LibreOffice/libreoffice-3.5.4+dfsg2/sc/source/core/data/ce
ll.cxx:1076
```

Right Setting up an advanced breakpoint in GDB to bypass LibreOffice's segmentation fault

inside a gdb session in order to figure out where exactly the segmentation fault happens. This is much easier than dealing with assembly code or memory addresses:

```
~ gdb /usr/lib/libreoffice/program/soffice.bin
(gdb) set args -o file_that_segfaults_libreoffice.
xlsx
(gdb) r
Program received signal SIGSEGV, Segmentation fault.
0x0000000200030000 in ?? ()
(gdb) bt
#0 0x0000000200030000 in ?? ()
#1 0x00007fffd6014225 in ScFormulaCell::Compile
(this=0x7fffe000a770, rFormula=..., bNoListening=false,
eGrammar=formula::FormulaGrammar::GRAM_NATIVE)
at /home/rene/Debian/Pakete/LibreOffice/libreoffice-
3.5.4+dfsg2/sc/source/core/data/cell.cxx:1076
```

As shown in the previous listing, you need to set up the program's parameters first (in this case, the document to be opened), and then execute it. Soon after, the program crashes and you get the segmentation fault error message inside your gdb session. You quickly realise that the address where the segmentation fault is triggered is not valid: **0x0000000200030000**. So some sort of memory corruption issue must be in order. Therefore, you need to retrace your steps and have a proper look at the program's back-trace to find out which function is in the previous frame. Using the **bt** command, you find out that it is the **ScFormulaCell::Compile()** method. You also know that the last executed statement is located exactly at line 1076 in the **sc/source/core/data/cell.cxx** C++ source file. Since this is an open source project, its sources are publicly available and so you can install them easily:

```
# apt-get source libreoffice
```

Now that you have the sources, you can see the whole **ScFormula::Compile()** method's implementation and look closely at its lines 1075 and 1076:

```
if(pCodeOld)
delete pCodeOld;
```

Apparently, the segmentation fault happens at some point in the program flow, after freeing the memory address pointed to by **pCodeOld** (a pointer to a structure of type **ScTokenArray**). It is common practice to inspect the program's data after a crash, so in order to achieve that you use the **frame** command to inspect the **pCodeOld** pointer at **ScFormula::Compile()** right after the crash, like so:

```
(gdb) frame 1
(gdb) p *pCodeOld
$38 = {<formula::FormulaTokenArray> = {_vptr.
FormulaTokenArray = 0x7fffc40d63c0, pCode =
0x22b7960,pRPN = 0x22b86d0, nLen = 30032, nRPN =
50189, nIndex = 32767, nError = 0, nRefs = 30048,
nMode = 13 '\r', bHyperLink = 196}, <No data
fields>}
```

After setting a breakpoint at line 1075, you find that the program doesn't crash *every* time it executes the delete statement. So you inspect the structure's values right after the program is interrupted at line 1075 in order to compare them with the ones firing the segmentation fault. Now, these values are 0. So, your reasoning should be something along these lines: 'what if I can trace how many times this segmentation fault should happen in order to avoid it?' Of course, you are not acquainted at all with the code in charge of parsing a Microsoft Excel XML spreadsheet, so you are just trying to find a way to circumvent this bug.

> "Even when you don't have access to the sources, a program can still be patched"

Gdb enables you to alter the program's data and pack a bunch of gdb commands to be executed as soon as a breakpoint is hit. Aided by these facilities, you will make use of the previous breakpoint to alter the program flow only whenever some values held by the ScTokenArray structure are greater than zero. After giving the problem some thought, you come out with this:

```
(gdb) set args -o file-that-fires-the-segfault.xls
(gdb) set pagination off
(gdb) b cell.cxx:1075
(gdb) set $hits = 0
(gdb) commands 1
> set $check = pCodeOld->nRPN
> printf "Check is: %d\n", $check
> if $check>0
> printf "Patching pCodeOld to avoid the crash ..."
> set $hits++
> set var pCodeOld=0x0
> end
> c
> end
```

As shown above, you set a gdb local variable called **hits** storing how many times the segmentation fault should happen. You add some commands to be executed by gdb itself as soon as the breakpoint at line 1075 is hit (**commands 1**). You choose the **nRPN** field as a checkpoint to infer whether the segmentation fault should happen (**check>0**), updating the value held by the hits variable accordingly and altering the pCodeOld program's pointer to be null (**set var pCodeOld = 0x0**). Now, recall that at line 1075 in the cell.cxx C++ source file, a check of this sort is made: **if(pCodeOld)**. So, it comes as no surprise that by setting the value pointed to by pCodeOld to **0x0**, the previous branch will not be taken and no delete statement executed. In case the nRPN field's value is less or equal to 0, the program flow will just continue normally.

So, you run the program from the beginning with this breakpoint set in place, this time being able to open the document. The hits variable reports two hits. Right after having the document opened, you save it using the native **ODS**

More than debuggers

A debugger is impressively powerful. By using it, we can exploit the software, even when it is proprietary. We can take advantage of a routine implemented in a program by setting up its environment (ie, stack and the like) and then calling it from a debugging session. Although it is a difficult technique, it is widely used in reverse-engineering malware or when analysing anti-copy protection technologies.

Double-free error

Although it could seem harmless, a double-free memory error can sometimes lead to a well-known vulnerability where the heap structure is altered in such a way that arbitrary code could be executed. The **glibc** heap protections that deter this vulnerability can be disabled by running: **export MALLOC_CHECK_=0**. This would have solved your first case, saving you some pain, but it is not recommended to disable these protections at all.

format, closing **LibreOffice Calc** afterwards. Finally, you try to open the ODS file in a new **LibreOfice Calc** instance, this time with no problems at all. You, and you alone, have successfully circumvented this issue!

Fixing a double-free error

It is time to move on; your next assignment concerns a proprietary piece of software: the ATI graphics card software installer. While installing the graphic card drivers on a GNU/Linux box, and right before the installer ends its execution, a double-free or memory heap corruption error is triggered and the process killed, which unfortunately prevents the installer from actually setting up the drivers.

The GNU library libc6 implements some basic protections to avoid memory corruption of the heap. In this particular case, the corruption happens because of a double-free pointer condition. You know that using the open source ATI driver is out of the question, because it is brand-new hardware and the only drivers capable of detecting the device properly are those of ATI itself. Therefore, you are meant to find a way to fix it.

The defective software is an ELF-64 binary included with the ATI installer package: setup.data/bin/x86_64/setup. The exact double-free memory corruption error is reported by the glibc6 itself at **0x013cebf0**. Apart from giving you the exact address that is being double-freed, you also have the entire back-trace of the installer's execution. This way, it is feasible to determine the buggy instruction offset inside the **ELF-64** binary:

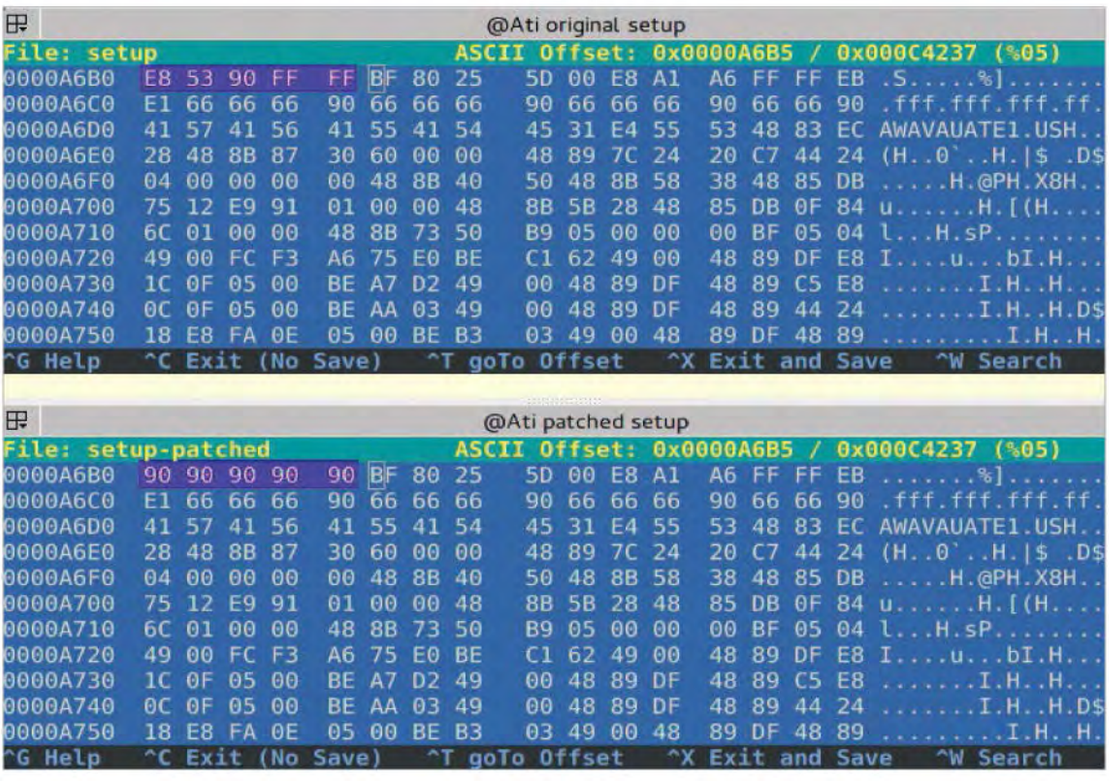
```
*** glibc detected *** .../setup.data/bin/x86_64/setup
double free or corruption (fasttop): 0x013cebf0.
=== Backtrace: ===
lib/libc.so.6(cfree+0x76)[...]
setup.data/bin/x86_64/setup[40a6b0]
```

According to the previous back-trace, a call triggering the double-free corruption error is placed at offset **0x40a6b0** in the setup binary file (last line). This time, though, the software is not open source, therefore you do not have its source code. But you are a bit aware of disassemblers and mnemonics, so you resolve to disassemble the binary using the **objdump** utility and look for the offset **0x40a6b0**:

```
~ objdump -d setup.data/bin/x86_64/setup > setup.S
~ cat setup.S|grep 40a6b0
8575 40a6b0: e8 53 90 ff ff callq 403708 <free@plt>
```

As clearly shown above, there is a call to the **free()** function at offset **0x40a6b0** (last line). And, as the previous back-trace has shown you, the instruction located in this offset is the one triggering the double-free or memory corruption error message. It seems pretty obvious to you that this instruction is freeing the already-freed address **0x013cebf0**, and that is a bug leading to a well-known software vulnerability (see the boxout, opposite).

As a good ICT expert, you want to corroborate this assumption by executing the program inside a gdb session. Because this program is based on the **ncurses** library, you have to redirect its output to another terminal, otherwise it would prove difficult to debug it properly:



Right Replacing the instruction **0xe85390ffff** with **0x9090909090** in the ATI installer (highlighted bytes)

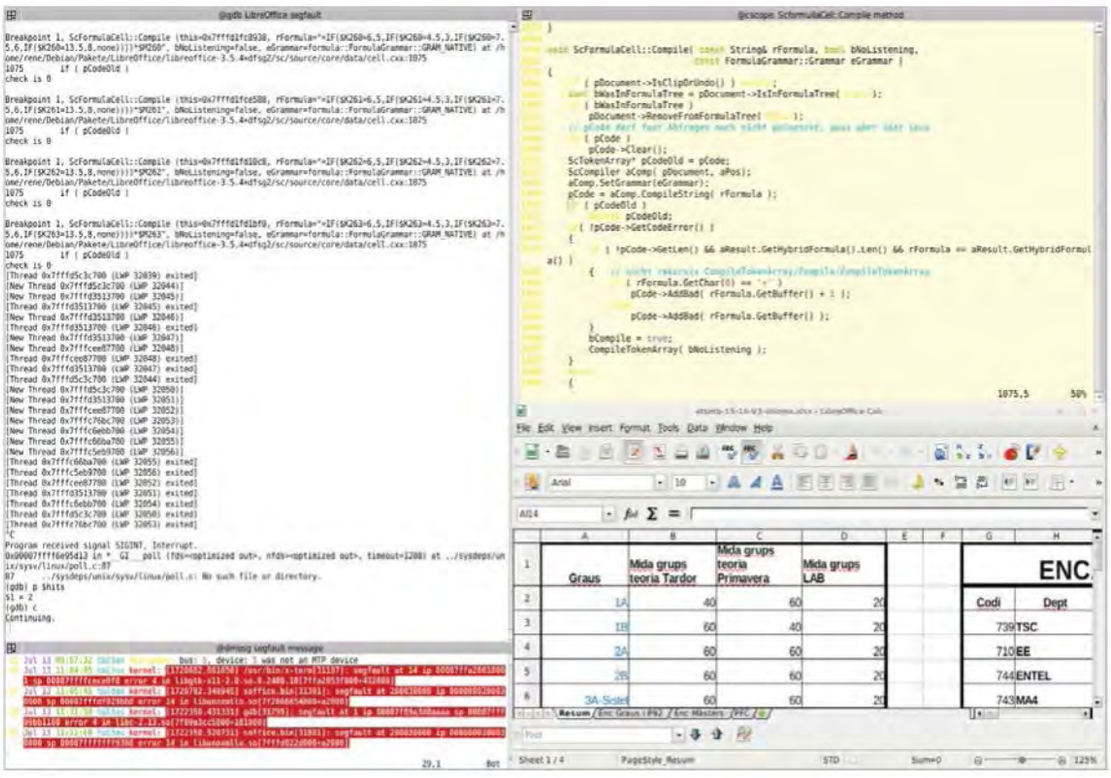
```
~gdb
(gdb) tty /dev/pts/12
(gdb) file setup.data/bin/x86_64/setup
(gdb) b *0x40a6b0
Breakpoint 1 at 0x40a6b0
(gdb) r
#1 0x40a6b0 in ...
(gdb) stepi
0x0000000000403708 in free@plt ()
(gdb) x/8w $rsp
0x7fff1787bb00: 0x00000000 0x00000000 0x7c0ba9e0
0x00007f3b
0x7fff1787bb10: 0x013cebf0 0x00000000 0x01318140
0x00000000
```

After redirecting the program's output to **/dev/pts/12**, you set a breakpoint at the offset address **0x40a6b0** (lines 2-4), where the buggy instruction **free()** is located. Then you run the program (line 6). Once the program flow reaches the buggy instruction, it stops. At that point, you execute just one machine instruction with the **stepi** command, analysing the stack before actually calling the **free()** function that irremediably leads to a double-free corruption error message (lines 10-12). Bear in mind that the stack holds the free function's parameters, so by running the **stepi** command you are allowing the stack to be set up properly before the program actually calls the **external** function. As you have previously seen, the **free()** function is freeing the address **0x13cebf0**. According to the previous listing there is, inside the stack, this very same address indeed (last line). So far so good – your statement has been corroborated and now an obvious conclusion is at hand: there is a double-free memory issue because the call to free at offset **0x40a6b0** is trying to free a previously freed pointer that was pointing, at some point, at address **0x13cebf0**. Now, an obvious question manifests itself: how can you fix it?

Well, you do not have the ATI installer's source code, but even if you don't have access to the sources, a program can be patched. You know that the buggy instruction is inside the **ELF-64** binary, so you guess that all you need to do is replace the op-code instruction **e8 53 90 ff ff** with another one. Our reasoning is like this: if we did not want the **free()** function to be called at that offset, what other machine instruction do we have to use? The first one that comes to mind is the **NOP (0x90)** instruction. Since the free call is five bytes in length, you have to replace it with five NOP instructions. Aided by a hexadecimal editor, you replace those five bytes with **0x90**. After that, you try once again and run the program. This time, as expected, the installer does not crash and the drivers can be installed!

Conclusions

It is commonly believed that a debugger is of no use to an ICT expert. Of course, not every ICT expert shares this opinion. We truly believe that most software-related issues can be fixed or bypassed by means of debugging them, and hope this guide has been engaging enough to sweep away the sceptics.



Left Our gdb session shows the total hits after applying the patch (top-left), while dmesg shows the segmentation fault messages (bottom-left)

Automate your audio in Linux

Automatically play your music at pre-determined times thanks to some inventive scripts

Resources
VLC
videolan.org/index.html

Listening to music is not really a chore. Modern media players have got to a point where you can just launch them, select a playlist and go. You've got full control, but what would be really cool would be to have your computer play your music automatically. Think of all the extra precious seconds you could save, or have your computer wake you up rather than an unreliable mobile phone, scheduling your music for specific times of day or having it play when you turn it on – that sort of automation is the dream.

We're in the business of showing you how to do cool things with Linux. In this tutorial we're going to show you how to automate and schedule different tracks and playlists with a variety of triggers and methods. Make your working day much more interesting by letting your computer handle your music rather than spending half an hour deciding what to listen to. Make sure you have VLC installed first, though – it's usually under the package vlc in the repos.

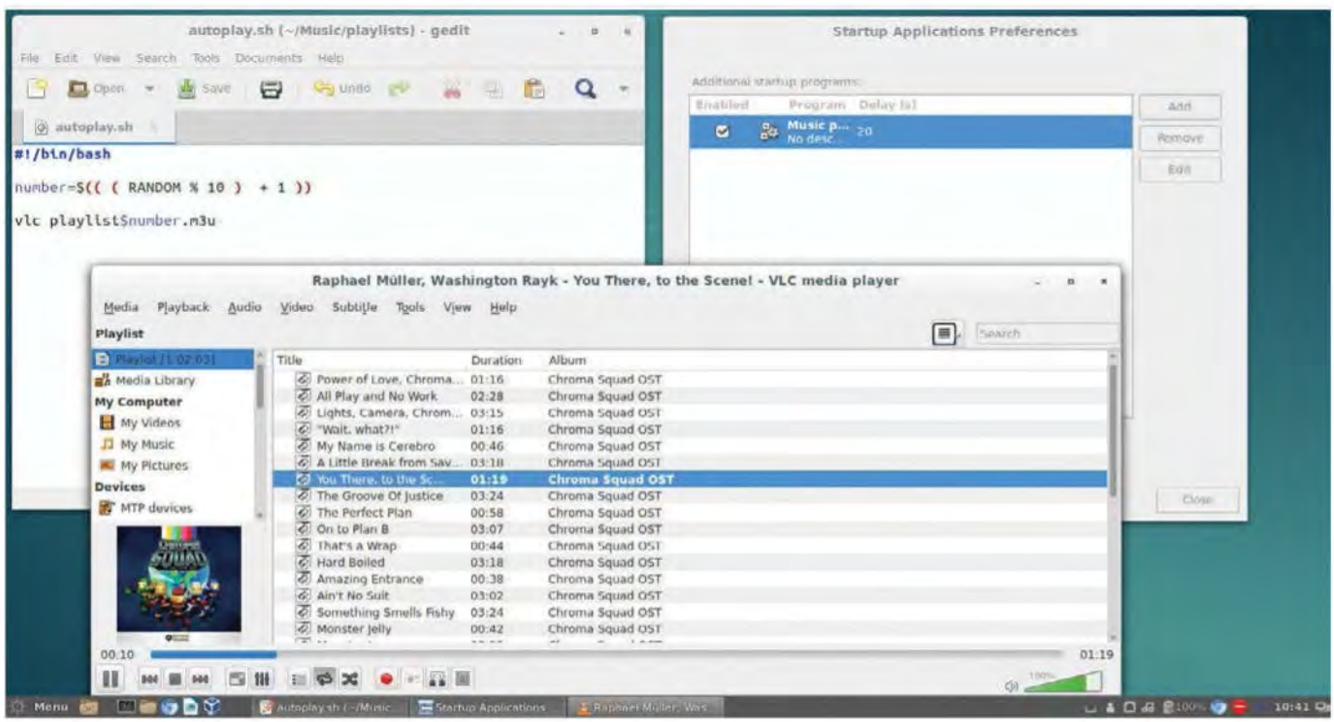
01 Understand the system

To get this to work, we're going to make extensive use of VLC's command line interface. This will just launch the VLC player with a few specific options and whatever file you decide you want to play. Very simply, you can play individual songs (and playlists) in the command line using something like:

```
$ vlc file.mp3
```

This will use the default setup of your VLC player, such as whether repeat is on or if it will stop what is currently playing in a VLC instance.

Below With a simple script and your own music library, you can automate your music listening experience



02 Build it up

For our automation, we're going to be making use of bash scripts, mostly to run the command line scripts that will play the music. This enables us to also add a bit more complexity in the bash script a bit later on in the tutorial in order to give you more choice for automation. These scripts can be activated on startup or by using a cronjob to schedule them throughout the day.

03 Basic script

We'll first of all build a basic version of the script we want to use. This will let us play an audio file in VLC player using the command line. Open up gedit, or whichever text editor you prefer, save a file somewhere as autoplay.sh and then enter the following code:

```
#!/bin/bash
```

```
vlc path/to/file.mp3
```

This simply lets the script know what to run itself as and then which command to execute. In this case, VLC will play the file.

04 Executable script

Though we've written the script, we still can't actually run it. This is because it doesn't have the right permissions to be executed as a program, but we can change this so it actually does. We use chmod to do this. You can elevate the permissions of your script with:

```
$ chmod +x autoplay.sh
```

> "Have your computer wake you up rather than an unreliable mobile phone – that's the dream"

You'll have to be in the directory that you saved it in. Stay in there as well for the next step.

05 Run the script

Let's test the script to make sure it actually works first. To do this, go back to the terminal and location we were in before and run the script with:

```
$ ./autoplay.sh
```

If all goes well, the VLC window will open and play the designated file for you. You can control the audio from the window or with any universal media keys, and closing it will kill the script as well.

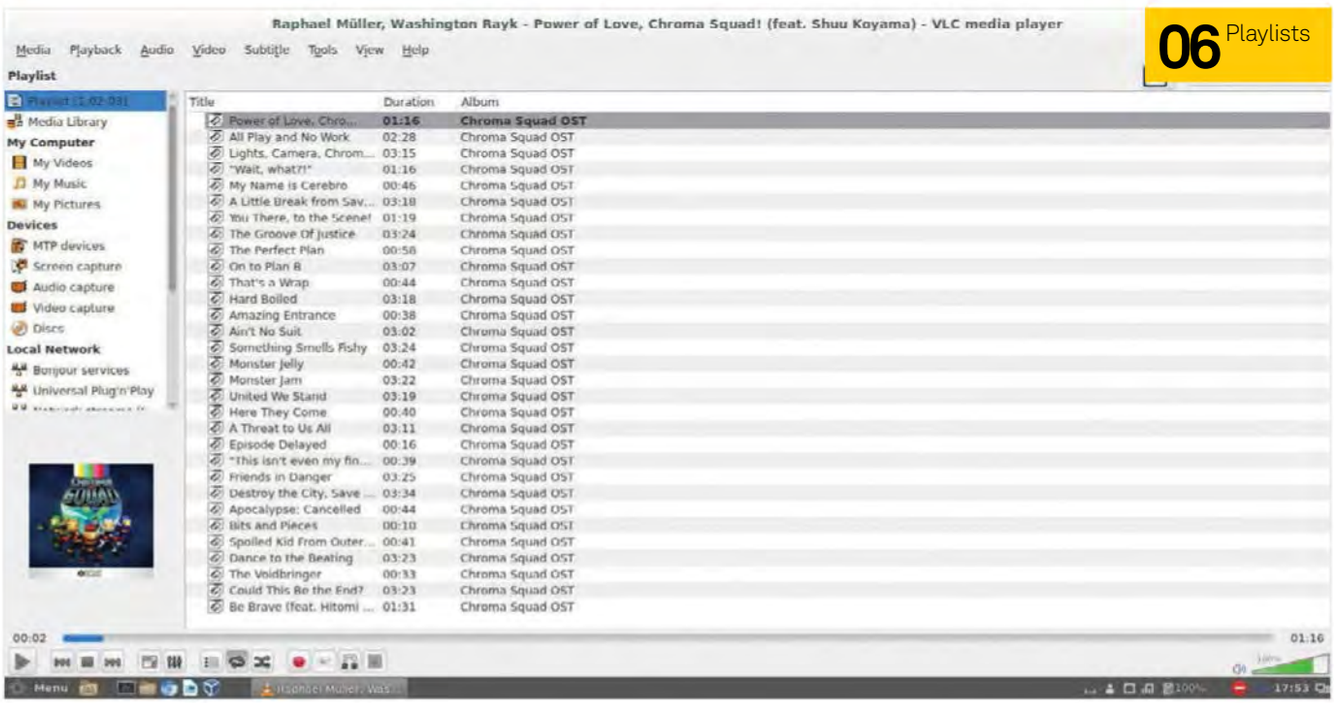
06 Playlists

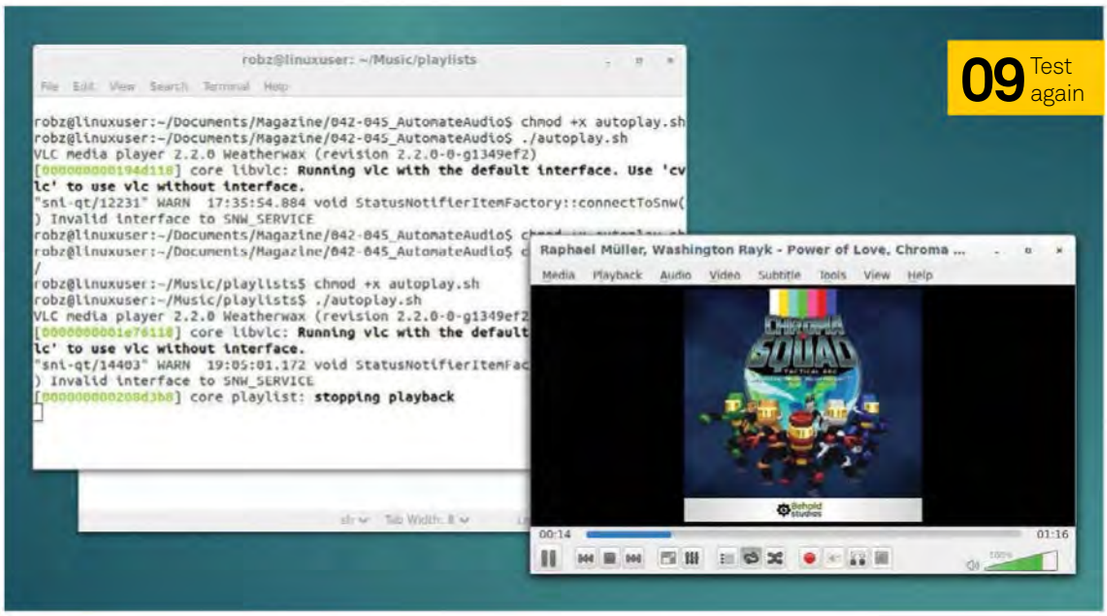
So now we understand how the script works, and can also prove that it does work, it's time to start making this automation better for day-to-day use. Playlists are the best way to do this: creating ten themed playlists to your own specification, which we can then choose at random and give to VLC to play. Create them in VLC if you want and then save them wherever you wish, or in the same directory as the script to make things easier. Name the files as numbers, from 1 to 10.

Other VLC options

We're using the basic VLC command to run the media player, but here are some options you can use:

-Z : Shuffle the current playlist
-L : Loop the playlist
-R : Repeat the first played item on the playlist
--spdif : Use spdif output if available





Right Any options you toggle in the VLC window will be remembered next time you play the file

Command line music

While we're running VLC using bash scripts on the command line, if you're using a CLI instead of a normal desktop environment, this won't work so well for you. However, there's CMus, a command line music player that can be used to play music from the command line. Switch out the **vlc** command for **cmus** in the script and it will work in the exact same way.

07 Random selection

So we've created ten playlists, each with a selection of music. How do we get the script to choose randomly between these playlists? We can use bash to create a random number in our script that can then be used to tell the vlc command which playlist to open. We'll do this using the $RANDOM operator, but limit it to only returning a number from one to ten.

08 The random script

The modifications to the script are fairly simple: we need to create a variable that we can give a random number, between one and ten, and then tell VLC to open that particular file (for example, 7.m3u). The script looks like this:

```
#!/bin/bash

number=$(( ( RANDOM % 10 )  + 1 ))

vlc playlist$number.m3u
```

The number variable is assigned the random number. The random integer is limited to ten, which would usually output zero to nine, so we add one to make it between one and ten.

09 Test again

Make sure everything's working by giving it a test – we do this again by running:

```
$ ./autoplay.sh
```

It should play the playlist in VLC without any issue. If you want to, now's a good time to hit shuffle and repeat to make sure it keeps playing the playlist, but not in the same way each time. Any changes you make to the playback here will be remembered for next time and any time after.

10 Scheduling

If you want to schedule the playlist, perhaps as an alarm or to mix up playlists during the day, you can create a cronjob that will run the script at certain intervals. We need to first construct the cronjob we want, though.

A cronjob takes into account the minute, hour, date, month, day and year, in that order. All of these are denoted by numbers, with days of the week going from zero to seven where both zero and seven are Sunday. So to have our script run at 8 AM as an alarm, we'd set the cronjob as:

```
0 8 * * * *
```

11 Create the cronjob

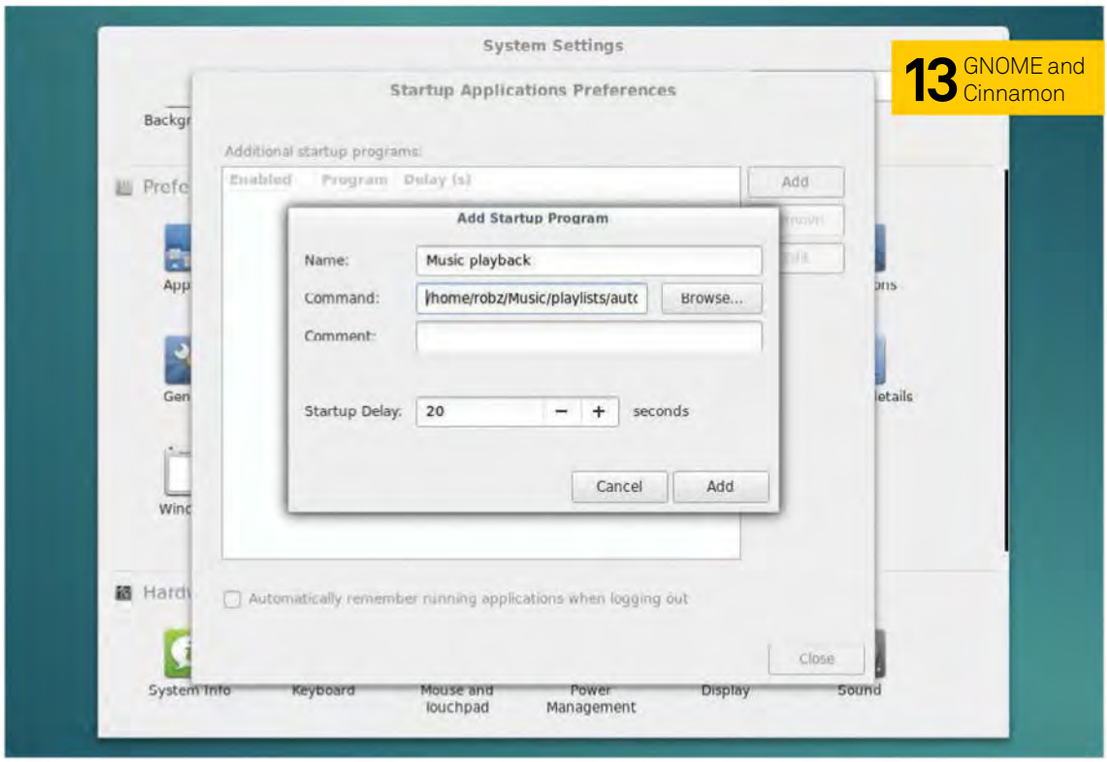
The crontab files, where all the different cronjobs are meant to be kept, can be accessed from the terminal using **crontab -e**. From here, scroll to the bottom of the commented section and add the line for your new cronjob. For the case of doing it at 8 AM as an alarm, use a line like:

```
0 8 * * * * /path/to/autoplay.sh
```

Save the file and then at 8 AM every day, if your system is on, a random playlist will start. Remember that the asterisks here denote wildcards, so they mean any date, any month, etc.

12 Startup script

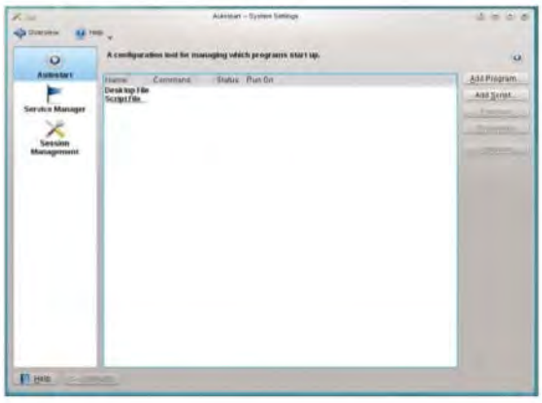
Scheduling music for certain times is great, but if you're not using your computer as an alarm, how can you make it so the music starts as soon as your PC turns on? Well, the script we've created can actually do this same function – we just need to call it differently. While Linux distros have their own startup mechanisms, since we want the script to run when everything is booted up, we should instead use the desktop environment's startup functions.



Left Your desktop of choice will have its own means of adding the script to startup

13 GNOME and Cinnamon

For any distro that uses a GNOME-based desktop environment, such as GNOME itself, Unity and Cinnamon, you can add scripts to the startup applications list. You can find this in the System Settings menu under Start Applications Preferences; from here click Add and find the script. Give it a name and comment if you wish, then log out and back in again to test it. We've given it a delay so that GNOME has a moment to properly load before trying to execute the command.



14 KDE

Adding a startup script to KDE is largely the same as GNOME, although you need to differentiate between a program and a script within the system. In the KDE System Settings, look for the Startup and Shutdown menu; the first tab is Autostart and this is where we need to put the script. Click on Add Script and browse to its location. Once you've found it, change the Run On option to Startup and save the rule. Again, you can test it by logging out and back in again.

> "Since we want the script to run when everything is booted up, we should instead use the desktop environment's startup functions"

15 Other desktops

Different desktop environments use different methods to launch apps at startup – most of them are similar to the method above, but with others you may have to edit config files to get them working. Your best bet is to look at the documentation for the desktops and see how you should go about setting the script to be run at startup.

16 Listen to music

With all this setup, your music listening experience should be just that little bit better throughout the day. Remember to mix up your playlists or add more every now and then. Also, don't be afraid to just turn off the playlist the script has chosen for the day and use your own. You can even use other media players if you want to take advantage of their different functions.

Set up game controllers to work on Linux

Try out Linux with a PlayStation, Xbox or other USB controller – perfect for media PCs

Resources

Desktop environment

Compatible USB game controller

Keyboard and mouse is the mainstay of GUI interaction. While we may laugh at the pictures of the very first mice from the Sixties, it's been the primary way we interact with graphical interfaces for a long time now, and even with the advent of touchscreen computing it doesn't seem like it's going away.

When counting touchscreen as an alternative, there are still other ways to control interaction on a graphical desktop. In this case, we're talking about game controllers. They are quite similar to a mouse in many ways – you move the point and press to interact. The moving part is a little different, of course, but it's functionally the same.

In this tutorial we're going to show you how to set up a USB controller to work with a Linux desktop environment, complete with clicks and extra button options for the controllers that have more than a few inputs. Having a setup like this can be good for many reasons, like having a low number of USB ports or a media PC, for example. We'll show you how to give it a try.

01 Universal controllers

To start with, we're going to look at making generic USB controllers and PS3 controllers work on Linux as mice. These all connect via normal drivers in the kernel, so all we need to do is change the settings with X to make them work.

> They are quite similar to a mouse in many ways – you move the point and press to interact

Above You'll need a USB or previous-gen console controller

02 X modifications

First of all, we need to install the specific joystick drivers that we want to use for X. Do this by opening up the terminal and installing the following package:

```
$ sudo apt-get install xserver-xorg-input-joystick
```

03 Get the files

We've created a basic configuration file that you can use to make your controller act like a mouse. It's geared around the PS3 controller for the moment, but we'll go into how you can modify it later. Download it with:

```
$ wget http://www.linuxuser.co.uk/wp-content
uploads/2015/05/50-joystick.conf
```

04 Move your files

We are going to copy over the file 50-joystick.conf from /usr/share/X11/xorg.conf.d/, so make a copy of it and then move the file we downloaded into the folder with:

```
$ sudo cp 50-joystick.conf /usr/share/X11/xorg.
conf.d/
```

05 Reboot to use

For everything to take effect properly, give your system a reboot. Load back into your desktop environment and plug your controller in. The PS3 sticks will both move the mouse in the same way and there are various right- and left-click buttons assigned, as well as a few Escape buttons.

06 Check controller inputs

If you need to change the inputs because your USB controller isn't quite working with the PS3 setup, then we'll have to figure out which buttons are which on the controller. Install the joystick test software with:

```
$ sudo apt-get install jstest-gtk
```

07 Test inputs

Still in the terminal, load up the tester with **jstest-gtk** while the controller is plugged in. You'll be able to see how the calibration of any analogue inputs is handled (which for a PlayStation controller is a lot), as well as which button number is activated when pressed.

08 Key bindings

Once you've figured out which buttons identify as what, you can now see which key codes relate to keyboard presses. Still in the terminal, find this out by using:

```
$ DISPLAY=:0 xmodmap -pk >all_keys.txt
```

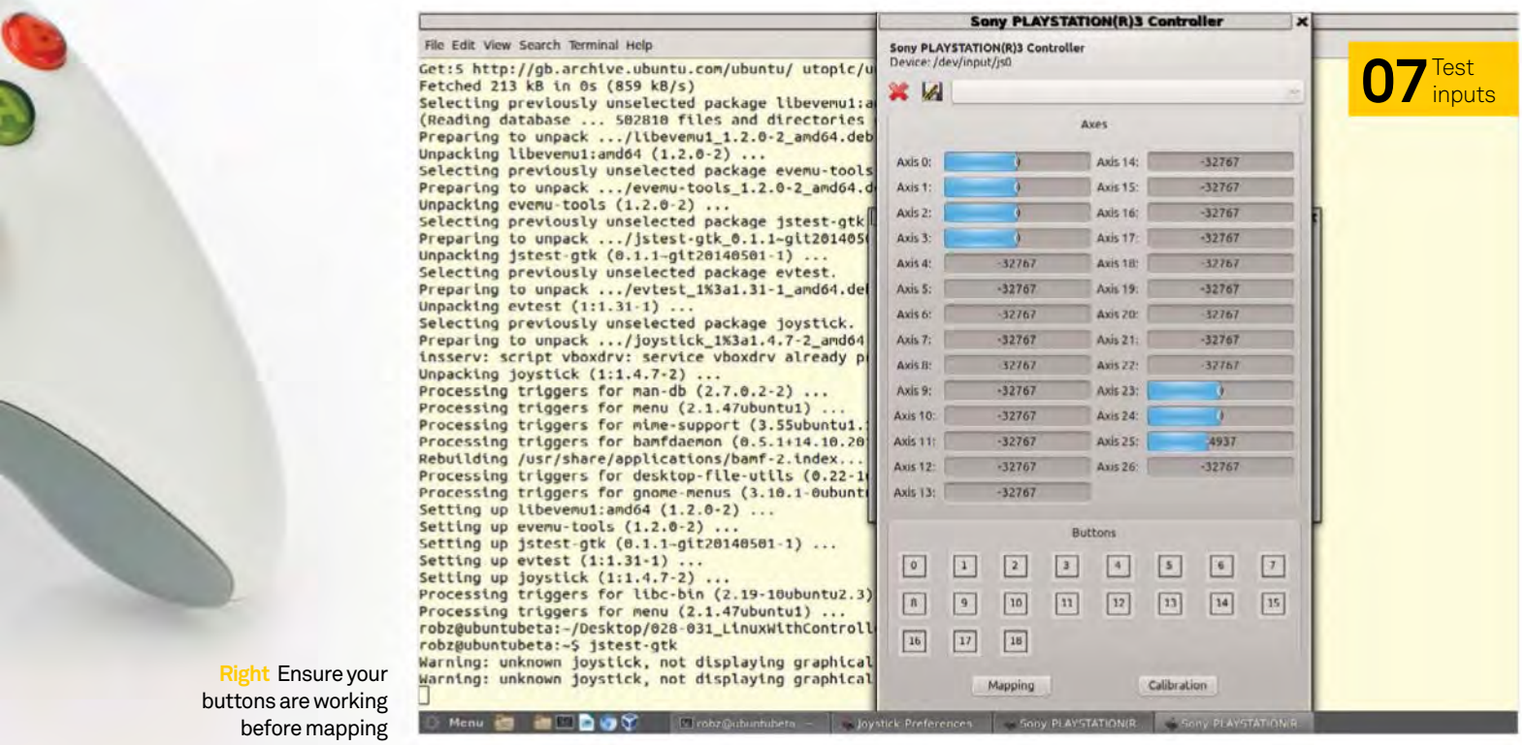
... and then open the file.

09 Read the file

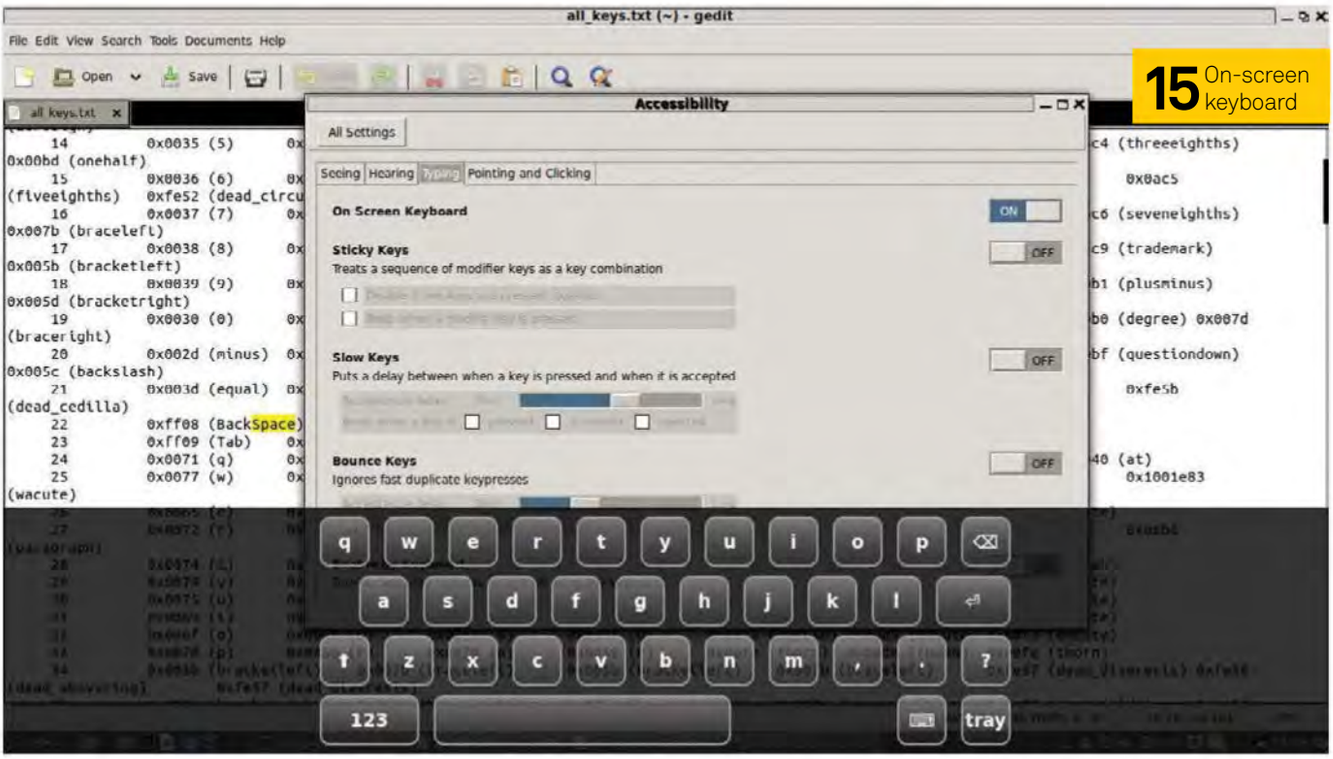
The keylist can be tricky to read, but all you need to do is pinpoint exactly what key you want to use. It is best to use gedit's find function to look for it and then make a note of the keycode value at the beginning of each paragraph so you have it recorded.

New PS4 controller

With new games consoles comes new controllers, and the Xbox One and PlayStation 4 have new versions of their classic pads. The DualShock 4 on the PS4 in particular is an interesting update, with a touchpad on the front. The controller itself is not properly supported in the kernel like the PS3 pad is, but development is under way to add extra drivers for it. You can find out more details here: **github.com/chrippa/ds4drv**.



Right Ensure your buttons are working before mapping



15 On-screen keyboard

Above The on-screen keyboard takes a little getting used to but is a decent replacement

10 Edit the config file

The file 50-joystick.conf is where the settings for what each button corresponds to are located. We assign each button to an action and this can be any keyboard key. Open it up in gedit or nano to start modifying how the buttons work.

11 Label the buttons

What we've done in our file is label what each button corresponds to as a comment. We've also put which key it relates to, as it can help to identify exactly what the setup should do. We recommend that you give this a go.

12 Add a space

In the current setup, triangle on the PS3 controller is free. If you plan to use the controller to replace the keyboard a little as well, we can try and have the triangle be space. In the key list, you will see that space is listed as keycode 65.

13 Program the button

We have labelled it on the original file, but triangle is otherwise "MapButton13" on the joystick config file. Now we need to change the line so it makes it a space, so do this by editing it to become like the following:

```
Option "MapButton13" "key=65"
```

14 Include more buttons

In the original code, there are a lot of reused buttons to make it as easy as possible for people to pick up the mouse aspect of the pad. This means there's a lot that you can reassign and that are spare in general. For example, change circle to be backspace (keycode 22) instead of escape.

15 On-screen keyboard

If you're using the controller as a full replacement for a mouse and keyboard, you can bring up an on-screen keyboard to deal with the removal of the physical keyboard. This can usually be found in the accessibility options for your desktop. Once turned on, you can then use the mouse function of your pad to click on the specific keys and use the space and backspace that we set previously to make it slightly easier for you to carry out.

360 Controller

LS (move)
Move cursor
LS (click)
Backspace
RS (move)
Mouse scroll
RS (click)
Space
D-pad
Up/down/left/right
Reset
Alt+left
Start
Alt+right
LB
Page up
LT
Volume down
RB
Page down
RT
Volume up
Y
Enter
A
Mouse left-click
B
Mouse right-click
X
Mouse middle-click

> "The file 50-joystick.conf is where the settings for what each button corresponds to are located"

16 Use an Xbox 360 pad

Using a 360 controller on Linux is a slightly different process as it uses its own specific driver set. While it is different (and specific to the controller), there are also upsides as you get a little more control over what you can do.

17 Add new drivers

Open up the terminal and install the xboxdrv controller drivers so we can actually use the 360 controllers on the system (although they will work with original Xbox controllers if you have a USB adapter):

```
$ sudo apt-get install xboxdrv
```

18 Quick mouse

Once it's installed, you can plug in your controller and quickly activate it to use as a mouse with the built-in tools by using the following two commands:

```
$ sudo rmmod xpad
$ sudo xboxdrv --mouse
```

19 Default mouse controls

By default, the controller will have a series of mouse and keyboard functions. The left stick is used for the mouse pointer, while the A button is used for left-click and the B button for right-click. Y is enter, the click on the left stick is backspace, the click on the right stick is space and there are a few more assigned by default.

20 Modify the buttons

Even though this is the default setup, you can easily change it as you wish – you'll have to do it on the command line when you first setup the mouse by adding extra options to the command. The setup is much more complicated than the PS3 controller, although you get much more control over it. Refer to the xboxdrv documentation on how you can make use of them: **bit.ly/1yRwy4b**.

21 Custom script

If you want the 360 controller to work on startup, you'll need to write a shell script and have it activate during startup – otherwise you will have to manually enter the command every time you turn your system on.

22 Control your PC

With these tools you can start properly controlling your system using these game pads. Use it in your living room, on your main system, or wherever is the most convenient for you. Soon you should be able to do it with PS4 and Xbox One controllers as well. ■

Above You can use multiple controllers by running multiple instances of the xboxdrv driver

Speed up your system by combining SSD and HDD

Make a 'best of both worlds' scenario for your system with a hybrid storage environment

Resources

SSD hard drive (minimum of 64 GB)

HDD hard drive

LuckyBackup
bit.ly/1L6bifD

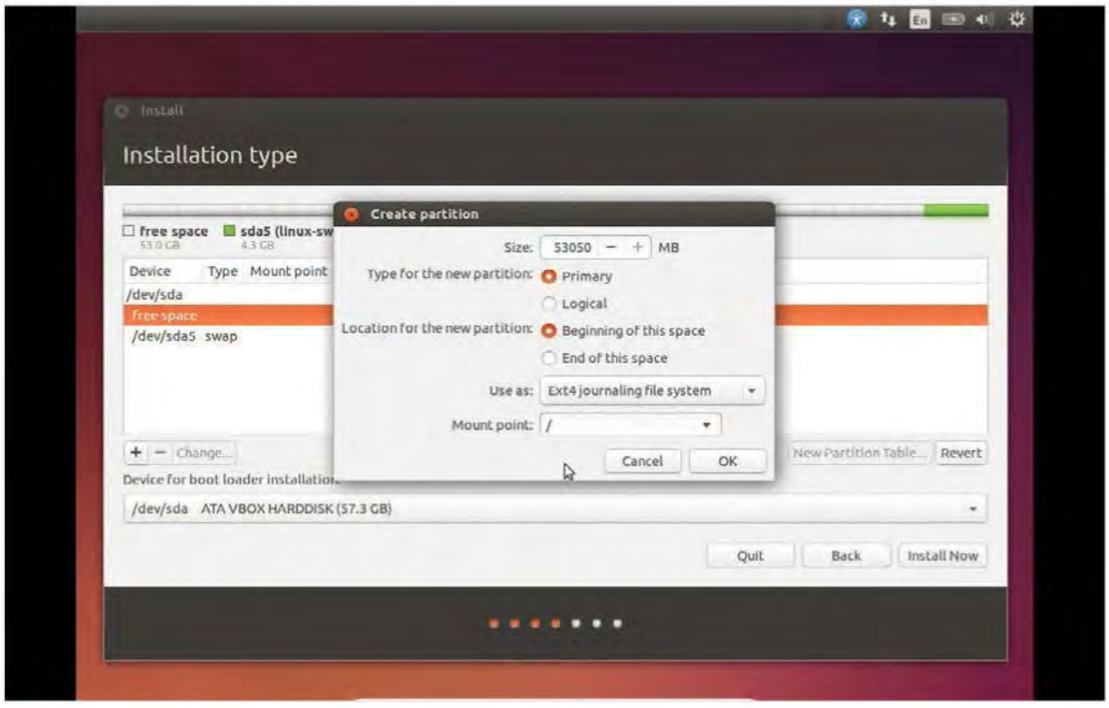
It's been a while now since SSD drives became a thing in modern computing. The price for reasonably-sized drives is now affordable, especially compared to the hundreds you would have spent on 64 GB a few years ago. Large amounts of storage space on SSD can still be a bit pricey, though, with standard hard disk drives giving you far more bang (or in this case, gigabytes) for your buck. There's also the problem of SSDs having a limited number of times they can be written to, at least compared to HDDs.

A combination of both, then, seems like a pretty good idea. In fact, it's one of the best ways to set up your system – using one drive for all the system files and another for all your documents and media files. The latter can invariably take up more space, so giving them the cheaper storage alternative makes more sense, especially as it doesn't quite require the speed of the SSD to work as well. Read on to find out what you need to be looking out for to create your perfect hybrid system.

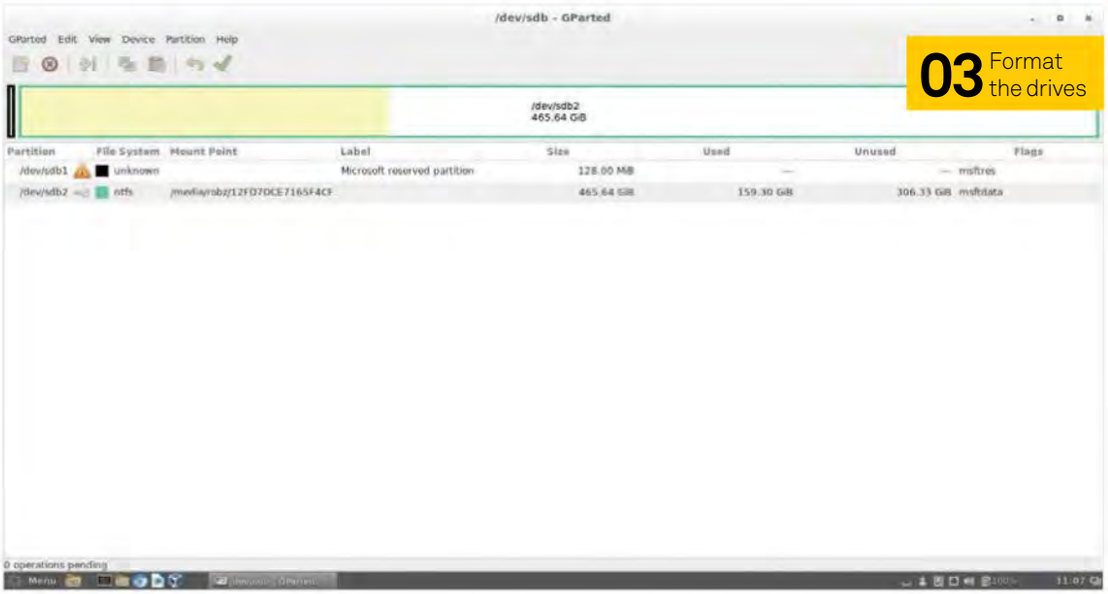
01 Choose an SSD

Your choice of SSD drive should take into account a few factors, but one of the most important is the required space. As we're treating the SSD as the install hard drive, it needs to be able to contain the entire operating system/Linux distro along with plenty of room for any software and packages you plan to install. As discussed, the size of SSD drives are much larger now, but the price increases quite sharply the larger you go.

Luckily, Linux distros usually require a lot less space than the offerings from the Church of Gates and the Cult of Jobs. 20 GB is usually ample space to install the distro, along with some resource-heavy desktops (bear in mind that the images take up a lot of space), so something like a 64 GB hard drive would technically be enough space. Still, 120 GB drives are fairly common and reasonably priced now as well, so we'd recommend going for one of those to make sure it has some degree of future-proofing.



Right The SSD will hold the swap and ext4 partitions for the distro install



03 Format the drives

Left Setting the HDD to NTFS means Windows machines on your network can access the files

02 Choose an HDD

This is the big hard drive in the equation, where all the files are going to be kept. How large you want it, we'll leave up to you, but we like to have at least 500 GB to 1 TB, preferably more, in our machines – you can go as small or as large as you want, though. If you have the space, you can even get multiple HDDs and make a RAID array, as they'll work just fine in conjunction with an SSD for the installation.

One thing we would suggest you look into, though, is a lower RPM hard drive. The standard 7200 RPM drives actually help increase read and write speed, which is useful for launching software and even boot-up itself. As we have those now on the SSD, which will have significantly improved speeds over 7200 RPM drives anyway, 5400 RPM drives are a good thing to consider. They take a small hit on read/write speeds for files, but they draw less power in the process and so produce less heat.

03 Format the drives

Before doing any installation, it's a good idea to format your new hard drive setup. Not every distro has full formatting tools built into the installation process so it can be handy to start it now.

The SSD we'll format as you would any other Linux hard drive. Using something like GParted, select the SSD drive and create a new partition. Put it at the end of the hard drive, make it a swap partition and give it the same amount of space as your system RAM. Fill up the rest of the space with an ext4 partition, which is where Linux will be installed to.

The HDD storage can be set up however you wish, but we suggest making it NTFS so it's a little bit more universal across systems. You can fill up the entire hard drive with a single partition as the swap and install are taken care of on the SSD. Set up the RAID at this point if you're doing that as well.

04 Install the distro

Now we can go ahead and install Linux. Boot into the live environment or installation prompt from your install medium, and begin the install process. When it gets to the part about setting up the file structure on the present hard drives and partitions, you'll need to do the following.

You need to set the SSD drive, which you'll be able to recognise as the one with the two partitions in both ext4 and swap, as the root location. Select the ext4 partition in the way your system allows and give it the option '/' for mounting. The installation will go here. The remaining drive and partition should be the larger HDD. Again, select it, and instead of the root ('/') mount point, we want to set it as the mount for home. This requires you to put the mount point as '/home'.

Allow the installation to continue as normal, and everything will be set up within the system when you go to boot it afterwards.

05 Use the setup

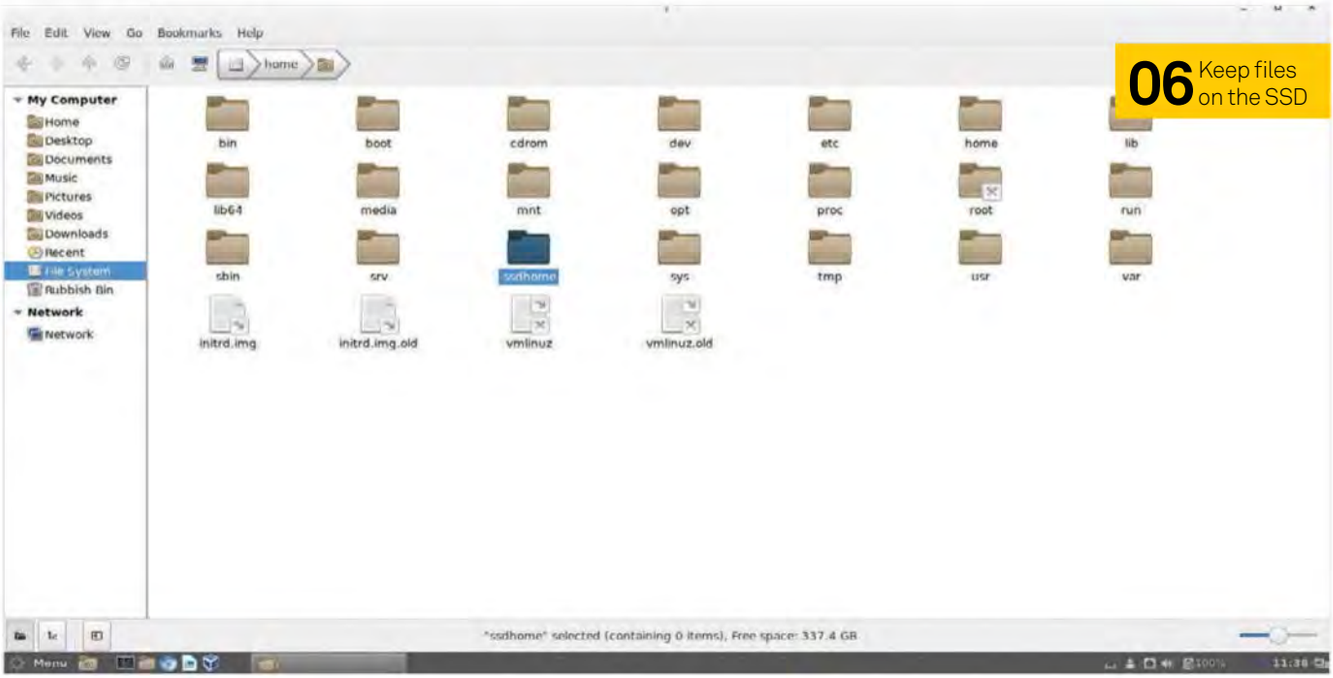
Actually using the setup we've created should be a nice and simple affair. The SSD is where all the boot, kernel and system files reside, so it should load a bit faster than previous, similar setups that didn't have an SSD. Installing and updating will be slightly faster, but limited by the download and CPU speeds. Any profile data, document, screenshot, video, etc will be either automatically saved into your home folder or put there manually by you.

An issue you might find is that if you set the storage drive to be NTFS, it may not quite be mounting properly on Linux. This can be easily fixed – installing the package ntfs-3g will allow you to work it. If it then needs to be mounted at start up, you'll need to add a line to the /etc/fstab file in the terminal. It would read something like:

```
/dev/sdb1 /home ntfs defaults 0 2
```

Hybrid drives

As well as our hybrid system, you can also specifically get hybrid hard drives. They have a small amount of solid state storage, backed up with much more disk storage. The idea is that it automatically puts files used more often on the solid state portion, to create the most optimal storage solution.



Above Got a folder full of music you play regularly? Stick that on the SSD too

06 Keep files on the SSD

If there are files you access regularly that could do with the improved speed of the SSD drive, you can always keep those on that drive as well. As you've got all this extra space, you might as well use it if you want to.

What we'd suggest is creating a separate folder on the hard drive to keep them on. In the terminal, create a new directory from root using:

```
$ sudo mkdir /ssdhome
```

... or whatever else you want to call it. On its own, the directory will work to keep files in, but you'll need admin permissions to do so each time, so we can tweak those permissions. From the terminal, use:

```
$ sudo chmod 755 /ssdhome
```

This enables people to read and write files into it on any access level, much like the normal home folder.

07 Check file activity

For the above, you may want to see how often a file is accessed and/or changed to assess whether or not you'd want to move it to the SSD folder. Linux is able to log this, but it's not set up by default (as depending on the number of users it can create a minor performance hit), but we can turn it on now to keep an eye out in the future.

We'll use auditd for this, which you can install with the auditd package from the repos for all major distros. Once installed, the service will start, and you can then add a directory or a specific file to track using:

```
$ auditctl -a exit,always -w [path to file or directory]
```

You can then check the logs to see when each file was accessed so you can make a decision based on that, and the logs are here: /var/log/audit/audit.log .

08 Move files automatically

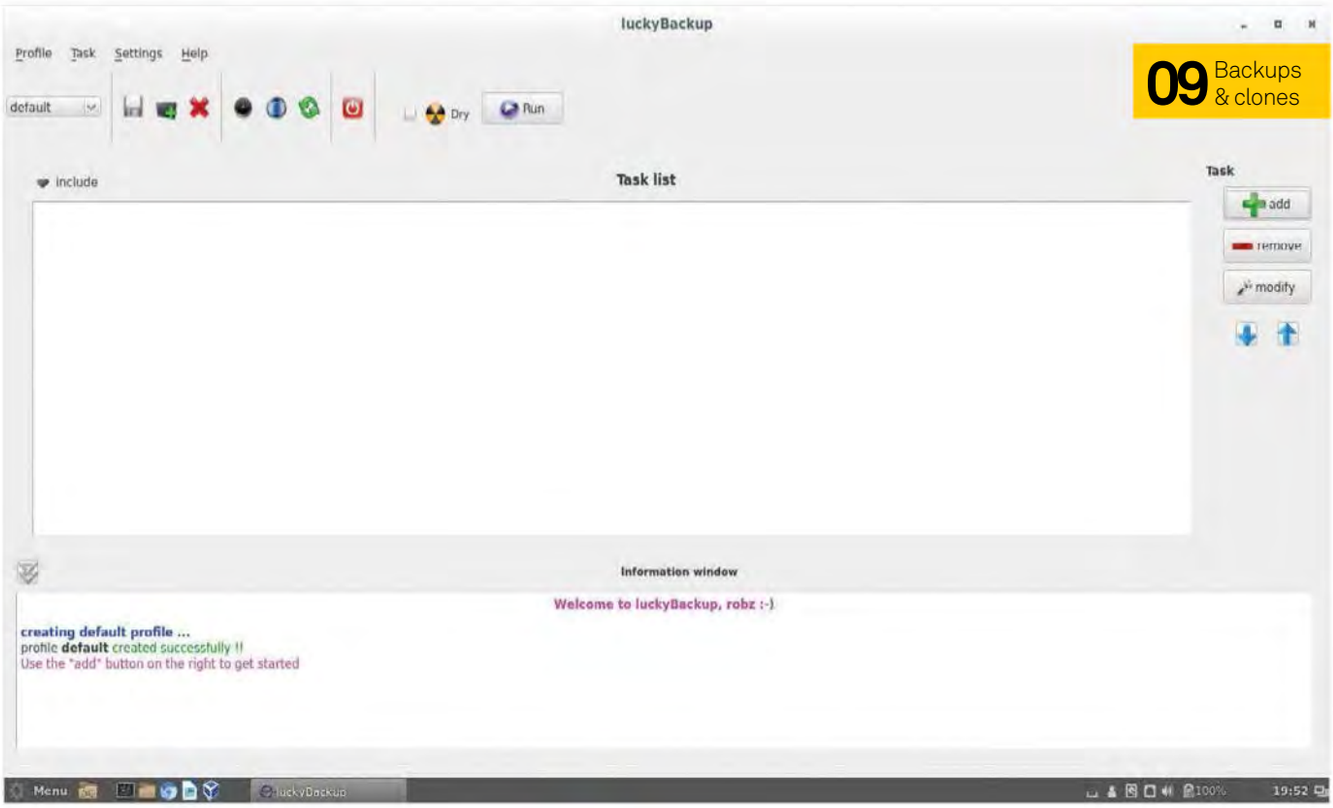
If any of the previous two steps are interesting to you, then maybe you'd like to think about moving a file between storage and the SSD drive automatically at certain times of the day, or specific days of a month, etc. We can set up a cron job so that the file is accessible when you need it most. First we need a bash script, like this:

```
#!/bin/bash
```

```
mv "/home/user/file" "/ssdhome/file"
done
```

Create an image

To back up your hard drive by cloning it, we like to use Clonezilla. It allows you to create an image of the hard drive that can then be reapplied by Clonezilla, and it only takes up as much space as the files do on the drive itself. For example, if you have 20 GB of files, the image will not be much larger.



Save it as filemove.sh and put it wherever you like. Elevate its permissions so it's executable by using **chmod +x**.

Now we can create a cron job that runs it whenever you want. In the terminal, enter crontab with:

```
$ crontab -e
```

Add a new line to run the script:

```
1 2 3 4 5 /home/user/filemove.sh
```

The numbers represent the minute, hour, day of the month, month and day of the week, and you can replace any one you don't want with an asterisk, which is the wild card.

09 Backups and clones

With a smaller hard drive as the actual installation partition, it's easier to back up the installation itself, while it's also easier to back up and important files you may want to keep from the documents folder.

With all the extra space on the other hard drive, any images you create of the install drive can be kept on there – that way, if the SSD dies, you can always replace it if need be. Assuming the amount of files on your storage drive are larger than the SSD, we don't recommend trying it the other way around.

In such a situation, local backups aren't very safe, though, as it's more likely to be the entire system that will have a problem. The SSD clone should be sufficiently small enough to keep in online storage, and you can always keep the most important files synced on the cloud. We like to use luckyBackup to back up files locally or across a network.

> "When it comes to a drive failure, replacing the hard drives is fairly simple in terms of getting the same dynamic set up again"

10 Replace drives

When it comes to a drive failure, replacing the hard drives is fairly simple in terms of getting the same dynamic set up again. If you have a RAID failure, you'll need to sort that out separately to the way the rest of the system works.

In terms of an SSD failure and replacement, all you'll need to do is reinstall the system to the SSD, or write the image back to it that we talked about cloning. If you have to do a fresh re-install, you'll need to follow along to Step 5 on how to make it mount at startup to the home folder.

In terms of replacing the HDD, it's much easier. You just need to format it, transfer any files back to it and make it mount on the home directory. You may have to update fstab as well with a new UUID.

Above Development for luckyBackup has stalled but it is still very good software

Programming in Linux

080 100 ways to master the command line

088 Start programming in C#

092 Use your Kinect with Linux

096 Intermediate AWK programming

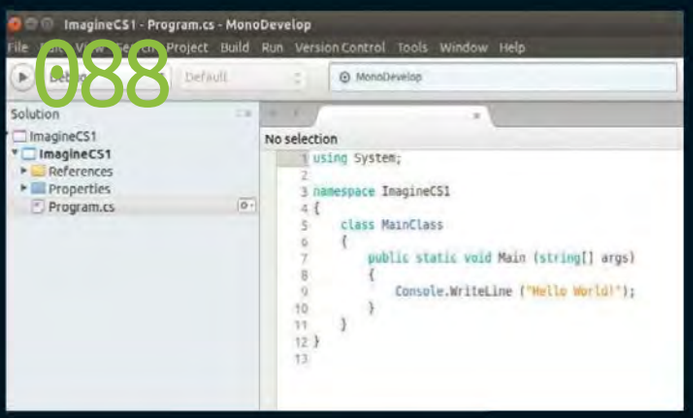
100 Launch your first Docker container

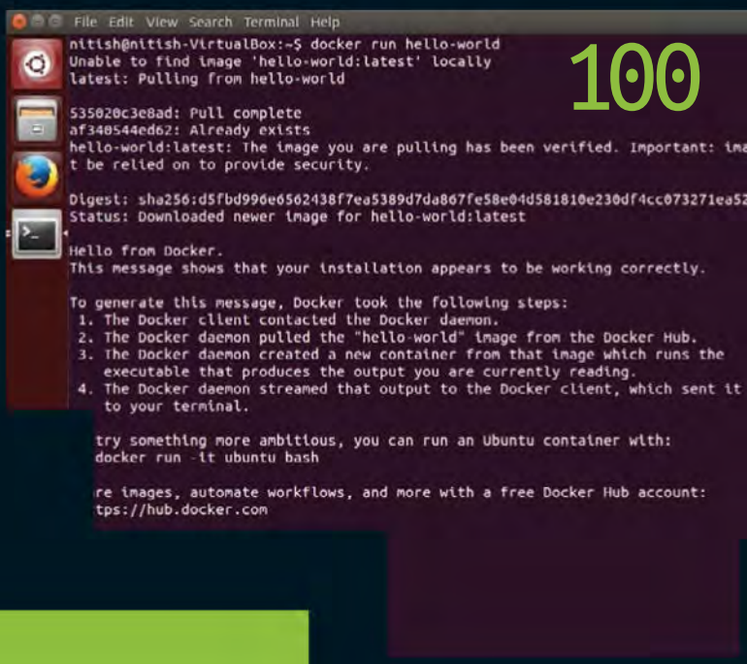
104 Deploy software to Docker containers using Deis

108 Run science experiments on the ExpEYES kit

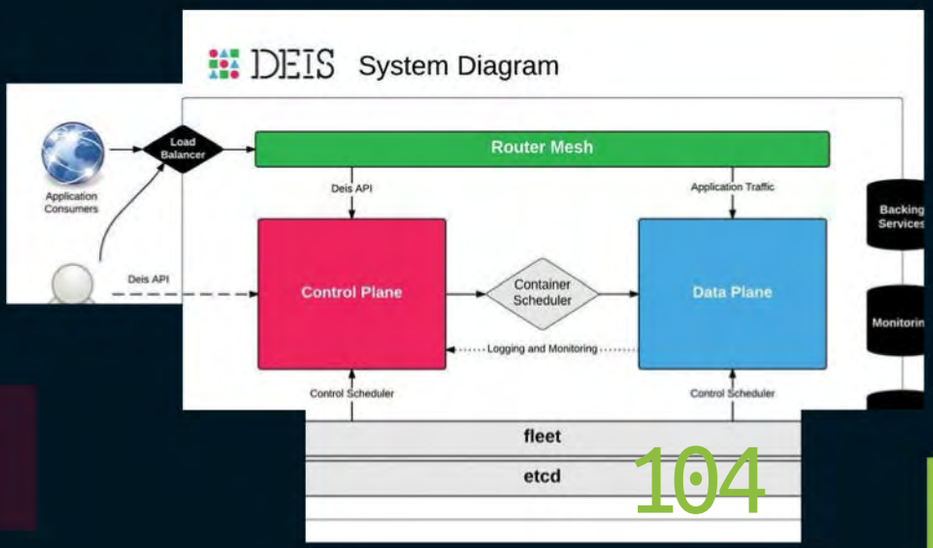
112 Program a Parrot AR.Drone's flight path

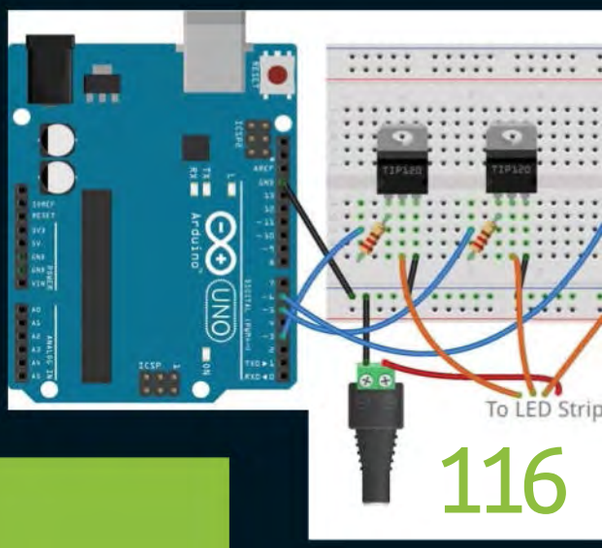
116 Home automation with your Pi

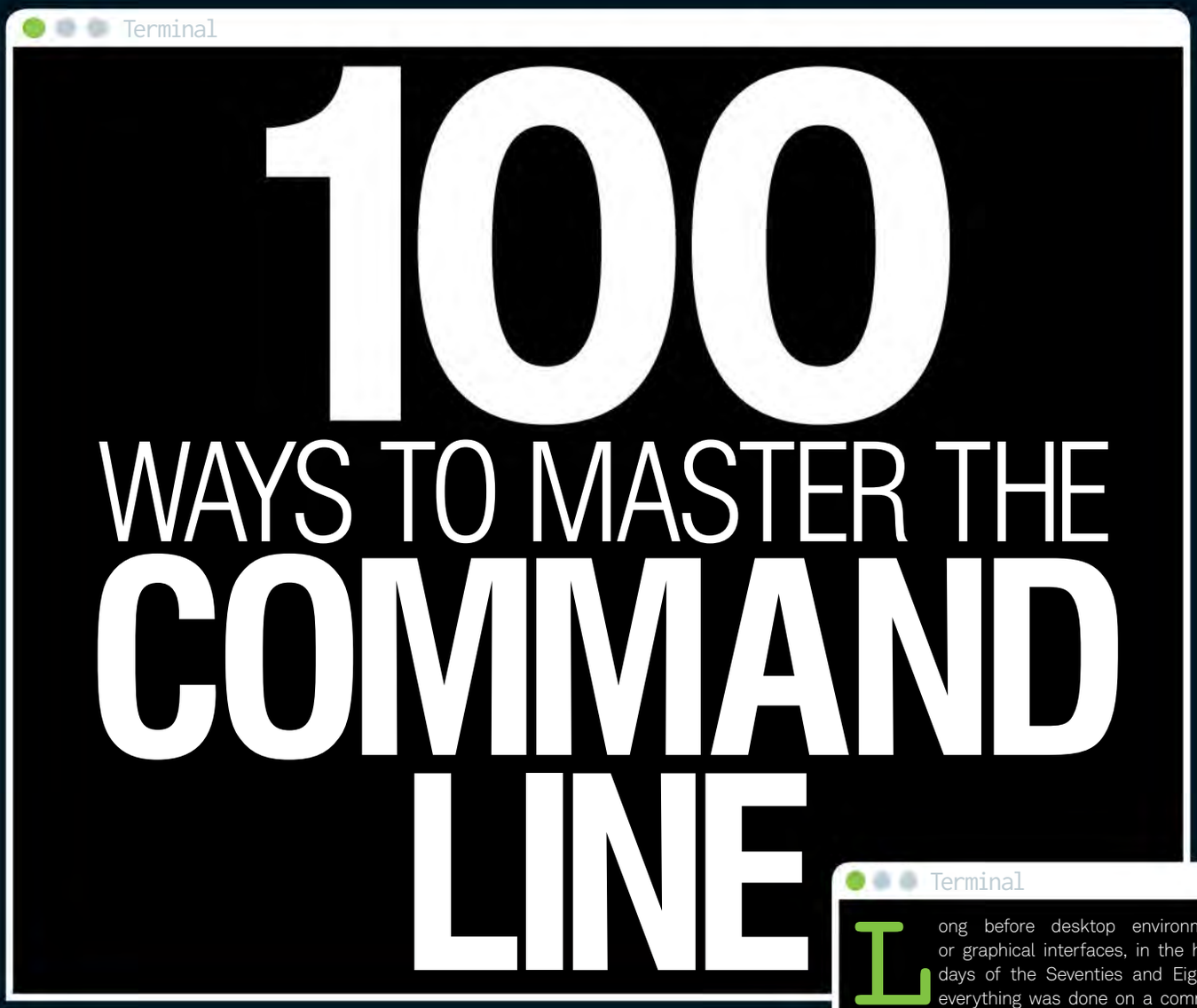





108


DEIS System Diagram
Load Balancer
Router Mesh
Application Consumers
Deis API
Application Traffic
Deis API
Control Plane
Container Scheduler
Data Plane
Logging and Monitoring
Backing Services
Monitorin
Control Scheduler
Control Scheduler
fleet
etcd
104


To LED Strip
116



100 WAYS TO MASTER THE COMMAND LINE

>_Conquer the command line with these essential terminal tips for controlling Linux

Terminal

Long before desktop environments or graphical interfaces, in the heady days of the Seventies and Eighties, everything was done on a command line. Computing in the Nineties, while generally dominated by graphical interfaces and mice, wasn't quite separated from it and computer users of a certain age will likely remember at the very least using the infamous DOS prompt.

Nowadays, even in Linux, you can spend your entire time in the desktop environment and never even touch the command line or terminal if you're using the right distro. Thing is, you'd be doing yourself a disservice by not learning how to make the most of the command line as it can be an extremely powerful tool – especially as a lot of graphical software will have command line controls that you can use as well. So put on your best Nineties outfit, give yourself a bad nickname and get ready to look like a movie hacker.

NEED TO KNOW

>_Get to grips with the terminal before you take on the advanced tools

>_001 ls

You can use **ls** to list the files and folders that are inside the directory you are in.

>_002 cd

The **cd** command enables you to move between directories on your system. Like the following, for example:

```
$ cd /home/user/
```

>_003 cp

The **copy** command, or **cp**, can be used to copy files from one location to another. To do this use the command below:

```
$ cp file /home/user/Desktop/file
```

>_004 mv

Similar to **copy**, **mv** instead moves the file to the new location, deleting the original file:

```
$ mv file /home/user/Desktop/file
```

>_005 rm

The **rm** command is for removing or deleting files and directories. You can use it like:

```
$ rm file
```

>_006 mkdir

You can create directories with the **mkdir** command using something like:

```
$ mkdir folder
```

>_007 nano

Nano is one of the programs that enables you to edit text in files – it's vital for working and editing in the command line. You use it like so:

```
$ nano file
```

>_008 Tab

The Tab key enables you to auto-complete various commands and file names, and double-tapping lets you list all of the objects that have a similar name. This auto-completion works with unambiguous file and command names. For example, 'fir' gives you 'firefox' because no other commands begin with 'fir'. If you get multiple hits, continue typing to help narrow the selection and then hit Tab again.

>_009 Up

Up on the keyboard has the very simple task of enabling you to pull up the last command that was entered, to run it again or edit it.

>_010 Copy text

If you're in the terminal, you may want to copy text output to use somewhere. To do this, you can use: **Ctrl+Alt+C**.

>_011 Paste text

If you've found a command online or need to paste some text into nano, you can enter any text on the clipboard with: **Ctrl+Alt+V**.

>_012 Open terminal

This trick works in a lot of desktop environments: a shortcut for opening the terminal can be done with: **Ctrl+Alt+T**.

>_013 sudo

This lets you do commands as the super user. The super user in this case is root and you just need to add **sudo** to the start of any command.

>_014 Access root

The super user, or root, can also be accessed by logging in as if from the terminal by typing **su**. You can enter the root user password and then run every command as root without having to use **sudo** at all.

>_015 Stop processes

If you have run a command and either it's paused, taking too long to load, or it's done its job and you don't need to see the rest of it, you can stop it forcefully by using either **Ctrl+C** or **Ctrl+Z**. Remember, only do this if the process is not writing any data, as it is not safe in this instance and you could find yourself in trouble.

>_016 Access root without password

If your regular user is in the sudoers list (ie so that they can use **sudo** in general to run things as root), you can access the su account by using **sudo su**. This uses your normal user password to access root terminal functions.To add a regular user to the sudoers list, you would need to first log in as the super user by using **su**. Then, simply run **adduser username sudo** (replacing 'username'). Be careful who you add to the sudoers list!

>_017 Search for hidden files and directories

The **ls** command can be used in more than just the basic way of listing all the files in a folder. To begin with, you can use it to list all the hidden items along with the normal items by using:

```
$ ls -a
```

>_018 Home directory shortcut

The home directory is located at /home/user/ in the absolute filesystem, but you can use the tilde (**~**) to signify the home directory when moving between directories or copying, or doing mostly anything else – like with the following, for example:

```
$ cd ~
```

>_019 Link commands together

If you want to do a series of commands one after the other, like updating and then upgrading software in Debian, for example, you can use **&&** to have a command run right after the one before. For example:

```
$ sudo apt-get update && sudo
  apt-get install libreoffice
```

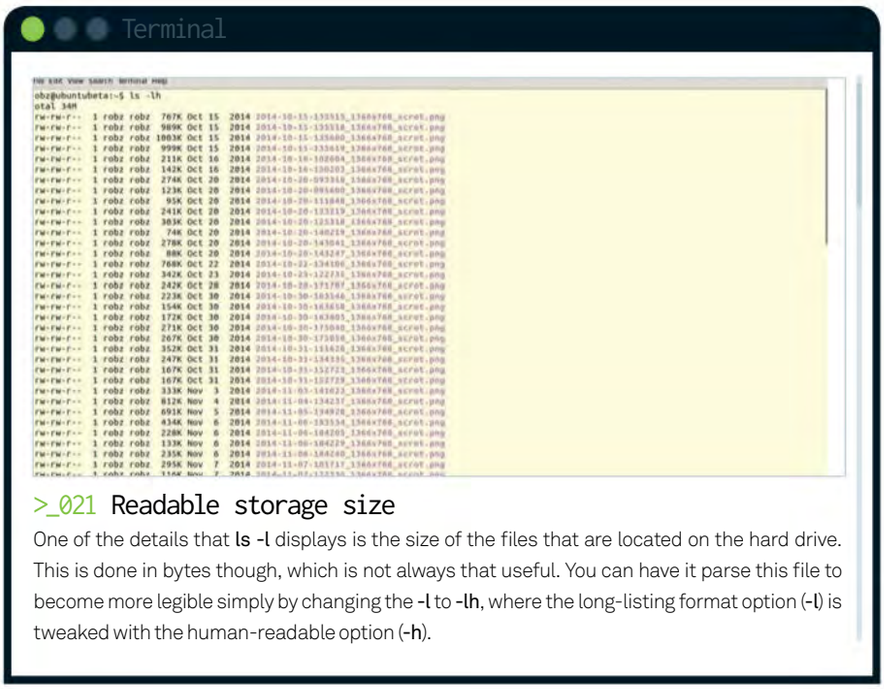
>_020 Long terminal input

Sometimes when using a list or anything else with a long terminal output, you might not be able to read it well. To make it easier to understand, you can send the output to another command, **less**, through the **|** pipe. It works like so:

```
$ ls | less
```

SYSTEM ESSENTIALS

>_Go a step further and learn how to control the terminal that bit better



>_021 Readable storage size

One of the details that **ls -l** displays is the size of the files that are located on the hard drive. This is done in bytes though, which is not always that useful. You can have it parse this file to become more legible simply by changing the **-l** to **-lh**, where the long-listing format option (**-l**) is tweaked with the human-readable option (**-h**).

>_022 Move to previous directory

If you want to move back to the directory that you were working on before, there is a **cd** command to do that easily. This one does not move up the filesystem, but rather back to the last folder that you were in:

```
$ cd -
```

>_023 Move up directory

The **cd** command can also be used to move up in the filesystem. Do this with two full stops, like so:

```
$ cd ..
```

>_024 General wildcards

The asterisk (*) can be used as a wildcard in the terminal to stand for anything. A typical use case is copying or removing specific types of files. For example, if you want to remove all PNG files from a directory, you **cd** to it and type:

```
$ rm *.png
```

>_025 More with the pipe

The pipe (|) can be used to feed all outputs into the next command, enabling you to call a piece of data or string from one command and put it straight into the next command. This works with **grep** and other core Linux tools.

>_026 Delete directories

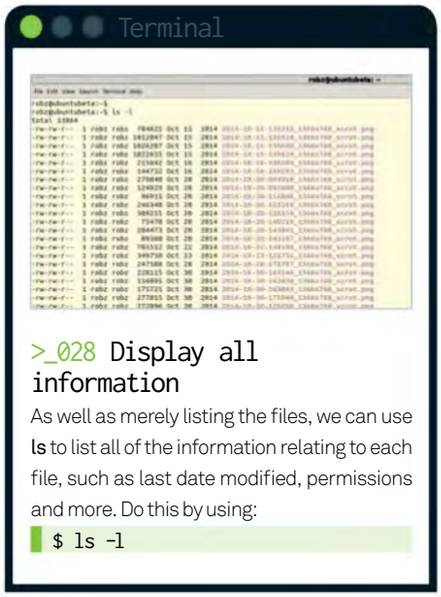
Using **rm** on a directory with objects within it won't work, as it needs to also delete the files inside. You can modify the **rm** command to delete everything within a directory recursively using:

```
$ rm -r directory
```

>_027 Shutdown command

You can shut down the system from the terminal using the **shutdown** command, the halt option (**-h**), which stops all running programs at the same time, and specifying a time of **now** so it turns off immediately rather than in 60 seconds:

```
$ sudo shutdown -h now
```



>_028 Display all information

As well as merely listing the files, we can use **ls** to list all of the information relating to each file, such as last date modified, permissions and more. Do this by using:

```
$ ls -l
```

>_029 Reboot from command line

Back in the day, rebooting required a slightly more complex shutdown command: **shutdown -r**. In recent years it's been replaced with a very simple:

```
$ sudo reboot
```

>_030 Timed shutdown

The timing function of the **shutdown** command can be very useful if you need to wait for a program or **cron** job to finish before the shutdown occurs. You can use the time to do a normal halt/shutdown, or even with **-r** for a reboot after ten minutes, with something like:

```
$ sudo shutdown -h +10
```

>_031 Log out

Logging out from the x session is generally advisable from the desktop environment, but if you need to log back out to the login manager, you can do this by restarting the display manager. In the case of many Linux distros, you use the command below:

```
$ sudo service lightdm restart
```

INSTALLATION

>_Managing your packages and updating your system is a key part of the command line

>_032 Debian: update repositories

Debian-based (and Ubuntu-based) distros use apt-get as the command line package manager. One of the quirks of apt-get as a package manager is that before upgrading or installing software, it does not check to see if there's a newer version in the repositories. Before doing any installation in Debian, use:

```
$ sudo apt-get update
```

>_033 Debian: install software

Unlike a graphical package manager or software centre, you can't quite search for the kind of packages you want to install, so you need to know the package name before installing. Once you do though, try:

```
$ sudo apt-get install package
```

>_034 Debian: update software

You can upgrade the software in Debian from the terminal by first performing the repository update command in Tip 32, followed by the upgrade command below:

```
$ sudo apt-get upgrade
```

>_035 Debian: uninstall software

As part of package management, apt-get enables you to uninstall software as well. This is simply done by replacing **install** with **remove** in the same command that you would use to install said package (Tip 33). You can also use **purge** instead of remove if you want to delete any config files along with it.

>_036 Debian: upgrade distro

Debian systems can often update to a 'newer version', especially when it's rolling or if there's a new Ubuntu. Sometimes the prompt won't show up, so you can do it in the terminal with:

```
$ sudo apt-get dist-upgrade
```

>_037 Debian: multiple packages

A very simple thing you can do while installing on all platforms is list multiple packages to install at once with the normal installation command. So in Debian it would be:

```
$ sudo apt-get install package1
  package2 package3
```

>_038 Debian: dependencies

Compiling differs between software and they'll each have a guide on how to go about it. One problem you might face is that it will stop until you can find and install the right dependency. You can get around this by installing **auto-apt** and then using it during configuration with:

```
$ sudo auto-apt run ./configure
```

>_039 Debian: force install

Sometimes when installing software, apt-get will refuse to install if specific requirements aren't met (usually in terms of other packages needing to be installed for the software to work properly). You can force the package to install even without the dependencies using:

```
$ sudo apt-get download package
$ sudo dpkg -i package
```

>_040 Debian: install binary

In Tip 39, we used **dpkg -i** to install the binary installer package that we downloaded from the repositories. This same command can be used to install any downloaded binary, either from the repos or from a website.

>_041 Debian: manual force install package

If the advice in Tip 39 is still not working, you can force install with **dpkg**. To do this you just need to add the option **--force-all** to the installation command to ignore any problems, like so:

```
$ sudo dpkg --force-all -i package
```

>_042 Red Hat: update software

Unlike apt-get, the yum package manager for Red Hat/Fedora-based distros does not need you to specifically update the repositories. You can merely update all the software using:

```
$ sudo yum update
```

>_043 Red Hat: install software

Installing with yum is very simple, as long as you know the package name. Yum does have some search facilities though, if you really need to look it up, but once you know what package you want, use the following command:

```
$ sudo yum install package
```

>_044 Red Hat: uninstall software

Yum can also be used to uninstall any package you have on your system, whether you installed it directly from yum or not. As long as you know the package name you can uninstall with:

```
$ sudo yum remove package
```

>_045 Red Hat: force install

The force install function on Red Hat and Fedora-based Linux distros requires that you have a package downloaded and ready to install. You can download things with yum and then force the install with:

```
$ sudo yum install --downloadonly
  --downloaddir=[directory] package
$ sudo rpm -ivh --force package
```

>_046 Red Hat: manual install

RPM is one of the package installers on Red Hat distros and can be used to install downloaded packages. You can either do something like in Tip 45 and download the package from the repos, or download it from the Internet and install with:

```
$ sudo rpm -i package
```

>_047 Red Hat: force manual installation

As in Tip 45, you can use RPM to force install packages if there's a dependency issue or something else wrong with any other packages that you have downloaded. The same command should be used as in Tip 45, with the **-ivh** and **--force** options present.

>_048 Fedora: distro upgrade

Yum has its own distribution upgrade command, but only in Fedora and they prefer you not to use it unless you have to. Nonetheless, you can use the fedora-upgrade package in yum with:

```
$ sudo yum install fedora-upgrade
```

> "You can force a package to install even without the dependencies"

SEARCHING WITH GREP

>_Search within files using the grep command to save time and find what you need

>_049 Search a file for a term

The basic use of **grep** is to search through a file for a specific term. It will print out every line with that term in, so it's best to use it with system files with readable content in. Use it with:

```
$ grep hello file
```

>_050 Check for lines

Looking for specific lines in a file is all well and good, but when you then start to hunt them down and you realise the file is hundreds of lines long, you can save yourself a lot of time by getting **grep** to also print out the line number. You can do this with the **-n** option:

```
$ grep -n hello file
```

>_051 Regular expressions

If you need to make a more advanced search with **grep**, you can use regular expressions. You can replace the search term with **^hello** to look for lines that start with hello, or **hello$** for lines ending in hello.

>_052 Wildcards and grep

When searching for lines, you can use a wildcard if you need to look for similar terms. This is done by using a full stop in the search string – each full stop represents one wildcard character. Searching for **h...o** will return any five-letter string with h at the start of the string and o at the end. Use it like so:

```
$ grep '\<h...o\>' file
```

>_053 More wildcards

You'll also be using wildcards to find something ending or beginning with a specific string but with no fixed length. You can do this in **grep** by using an asterisk (*) along with the dot. In the above example, we would have used **h.*o** instead.

DEVELOPMENT TIPS

>_Some terminal tricks for devs to help your command line skills become more efficient

>_054 Stop a system service

A system service is the kind of background software that launches at start up. These are controlled by the system management daemons like init or systemd, and can be controlled from the terminal with the **service** command. First, you can stop a service using:

```
$ sudo service name stop
```

>_055 Start a service

You can start system services that have been stopped by using the same **service** command with a different operator. As long as you know the service name, start it using:

```
$ sudo service name start
```

>_056 Restart a system service

This one is popular with setting up web servers that use Apache, for which restarts may be needed, along with other services that you customise along the way. Instead of running both the **stop** and **start** commands sequentially, you can instead restart services by using:

```
$ sudo service name restart
```

>_057 Know the ID

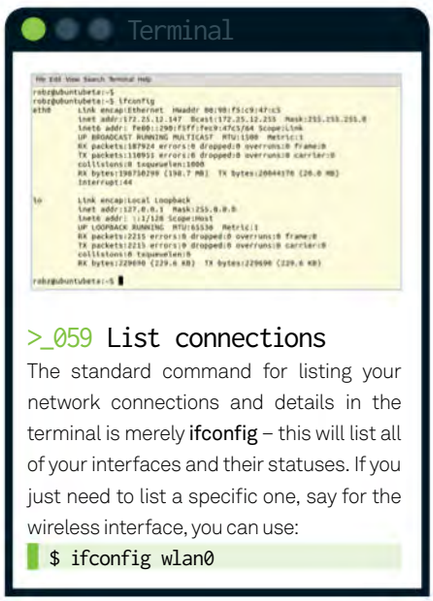
The ID that you can get for the software with **top** can be used to manipulate it, but a common reason to know the ID is so that you can end the process if it's having trouble stopping itself or using too many resources. With the ID in hand, you can kill it with:

```
$ kill 1234
```

>_058 Kill multiple IDs

Sometimes a process can be split up over multiple IDs (this is usually the case with a web browser – Google Chrome is a notorious example), and you need to kill multiple processes at once. You can either try and quickly kill all the IDs, or use the common name for the process and kill it with:

```
$ killall -v process
```



>_059 List connections

The standard command for listing your network connections and details in the terminal is merely **ifconfig** – this will list all of your interfaces and their statuses. If you just need to list a specific one, say for the wireless interface, you can use:

```
$ ifconfig wlan0
```

>_060 List USB devices

You may need to know which USB and USB-related devices are connected to a system, or find our their proper designation. You'll need to install it first, but once you have, you can use **lsusb** in the terminal to list all of the available devices.

>_061 List hard drives and partitions

Whether you need to check the designation of certain drives for working on them, or you just need to get a general understanding of the system's layout, you can use **fdisk** to list all the hard drives. Do this in the terminal with:

```
$ sudo fdisk -l
```

>_062 Check running software

Sometimes you'll want to check what's running on your system and from the terminal this can be done simply with **top**. It lists all the relevant information you'll need on your currently running software, such as CPU and memory usage, along with the ID so you can control it.

>_063 Unpack a ZIP file

If you've downloaded a ZIP file and you did it from the terminal or you're working from it, you can to unpack it using the **unzip** command. Use it like so:

```
$ unzip file.zip
```

>_064 Unpack a TAR file

Sometimes Linux will have a compressed file that is archived as a .tar.gz, or a tarball. You can use the terminal to unpack these or similar TAR files using the **tar** command, although you need the right options. For the common .gz, it's:

```
$ tar -zxvf file.tar.gz
```

>_065 Copy and write disks

Coming from UNIX is a powerful image tool called **dd**, which we've been using a lot recently for writing Raspberry Pi SD cards. Use it to create images from discs and hard drives, and for writing them back. The **if** is the input file or drive and the **of** is the output file of the same. It works like so:

```
$ dd if=image.img of=/dev/sda bs=1M
```

>_066 Create an empty file

Sometimes when coding or installing new software, you need a file to write to. You could create it manually with **nano** and then save it, but the terminal has a command similar to **mkdir** that enables you to create an empty file – this is **touch**:

```
$ touch file
```

>_067 Print into the terminal

The terminal uses **echo** to print details from files into the terminal, much like the C language. If you're writing a Bash script and want to see the output of the current section, you can use **echo** to print out the relevant info straight into the terminal output.

>_068 Check an MD5 hash

When downloading certain files it can help a lot to check to make sure it's downloaded properly. A lot of sites will offer the ability to check the integrity of the downloaded file by comparing a hash sum based on it. With that MD5 and the file location at hand, you can compare it with:

```
$ md5sum file
```

>_069 Run commands to x

Sometimes you need to do something concerning the x display, but the only way you can enter the command line is by switching to an alternate instance with **Ctrl+Alt+F2** or similar. To send a command to the main x display, preface it with **DISPLAY=":0"** so it knows where to go.

>_070 Create a new SSH key

When you need to generate a strong encryption key, you can always have a go at creating it in the terminal. You can do this using your email address as identification by entering the following into the terminal:

```
$ ssh-keygen -t rsa -C
"your_email@example.com"
```

>_071 System details

Sometimes you want to check what you're running and you can do this with the simple **uname** command, which you can use in the terminal with the following:

```
$ uname
```

>_072 Kernel version

As part of **uname**, you also get the kernel version. Knowing this can be useful for downloading the right header files when compiling modules or updating certain aspects. You can get purely the kernel version by adding the **-r** option:

```
$ uname -r
```

>_073 CPU architecture

If you're on an unknown machine, you might need to find out what kind of architecture you're running. Find out what the processor is with:

```
$ uname -p
```

>_074 Everything else

Uname enables you to display a lot of data that is available from the system and you can look at all of this information by simply using the **-a** option with the command:

```
$ uname -a
```

>_075 Ubuntu version

With all the distro updates you do, it can be tricky to keep track of which version of Ubuntu you are on. You can check by using:

```
$ lsb-release -a
```

> "To send a command to the main x display, preface it with DISPLAY=":0""

FILE PERMISSIONS

>_Learn how to view file permissions and then how to modify them properly

>_076 List file permissions

You can check the file permissions of every item, including hidden files and directories, in the terminal using **ls -la**. It will print out the file permissions as a ten-character string in the first column of output. The first character identifies the file type, with **d** indicating a directory and **-** indicating a regular file. We're interested in the last nine characters, which are actually three sets of three and are interpreted differently. For example:

```
rwxr-xr-x
```

R stands for read, **w** stands for write and **x** stands for execute. If they're present instead of a **-**, it means that it is present in that particular block of permissions. It's split up over three blocks: the first three being the user you are currently using, the second being the group and the third being for everyone else.

>_077 Change permissions

With the permissions ascertained, you can start editing them if you want via the **chmod** command. You edit each of the three permissions set by assigning it a number that treats the three-bit permissions set as a binary. So you'd do something like:

```
$ chmod 777 file
```

The first number is for the user permissions, the second is for the group and the third is for everyone else. The numbers mean:

```
7: read, write and execute, 111/rwx
6: read and write, 110/rw-
5: read and execute, 101/r-x
4: read only, 100/r--
3: write and execute, 011/-wx
2: write only, 010/-w-
1: execute only, 001/--x
0: none, 000/---
```

MEDIA CONTROLS

>_You can control your tunes while working inside the terminal

>_078 Pause music

Some audio players have command line controls they can use. Rhythmbox has this and it's a cool thing to use when you're stuck in the terminal and need to just pause your music for a moment. You can do this by using:

```
$ rhythmbox-client --pause
```

>_079 Skip music

The command line controls don't enable as much as you can get in the interface, however you can at least skip to the next track. Try it with the command below:

```
$ rhythmbox-client --next
```

>_080 Pause video

You can use Mplayer to launch video from the command line to watch. It's good for testing a video with different settings that you can affix to the video-playing command. What you can also do is control the playing video with keys – specifically, you can pause by using the space bar.

>_081 More video control

Mplayer gives you a few more controls while playing in the command line. You can use Enter to skip to the next item in a list, and otherwise, you can stop playback by using **Ctrl+C** to end the process entirely.

BEST OF THE REST

>_All the other commands that you might want to know for future reference

>_082 Open files in terminal

If you've got a file you can see in a graphical file manager, instead of opening a terminal and navigating to and then executing the file or script, you can usually run it directly from the terminal. To do this you usually just need to right-click and select a 'run in terminal' option.

>_083 Find files in terminal

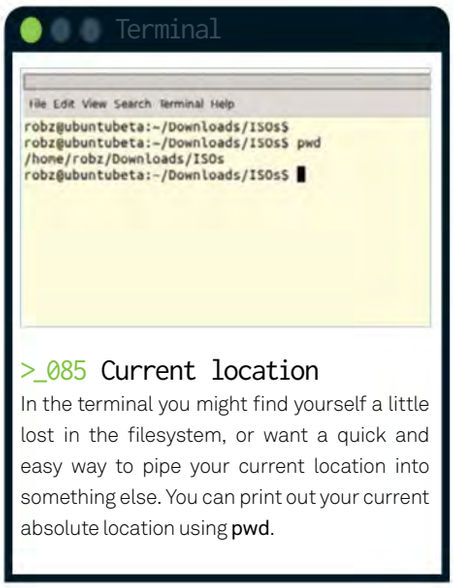
You can search for specific files throughout the filesystem by using the **find** command. You need to give find a location to search in and a parameter to search for. For simply searching for a file from root with a specific name you can use:

```
$ find / -name file
```

>_084 Locate files in terminal

Similar to find is **locate**, a newer tool that works slightly differently to find. While find also has ways to search for files by age, size, owner and so on, locate only really uses the name to locate, however it can do it so much faster. Use it with:

```
$ locate file
```



>_085 Current location

In the terminal you might find yourself a little lost in the filesystem, or want a quick and easy way to pipe your current location into something else. You can print out your current absolute location using **pwd**.

>_086 Move directories

When moving between directories you might want to be able to quickly return to the previous one that you were using. Instead of using **cd** for moving to the directory, you can instead use **pushd** to move and create a 'stack' of directories to move between.

>_087 Move back through pushd

Following on from Tip 86, once you want to start moving back up the stack to the first directory, you can use **popd** in the terminal. You can also check which directories you have stacked up by using the **dirs** command as well.

>_088 Process priorities

CPU priority for processes can be seen as running from -20 for highest priority or +20 for lowest. Putting **"nice -n X"** in front of any command enables you to change the priority from 0 to whatever number X is standing in for. Only **sudo** or root can elevate the priority of a process, but anyone can set one down the priority chain.

>_089 Download via the terminal

If you need to download something via the Internet in the terminal, you'll want to use the **wget** command. It will take any URL you specify and download it directly into your current location in the terminal (as long as you have permission to do so). Use it like:

```
$ wget http://example.com/file.zip
```

>_090 Change image formats

Instead of loading up an image editor like GIMP, you can actually change images in the terminal using the **convert** command. You can very simply use it to change the filetype or even change the size. Do it with:

```
$ convert image.jpg image.png
```

>_091 Alter image settings

As well as **convert** there's also **mogrify**, which is part of the same software package. You can use it scale, rotate and do more to an image more easily than with some of the **convert** commands. To resize you can use something like:

```
$ mogrify -resize 640x480! image.png
```

>_092 Send message to displays

This is an old school prank that can actually have good uses when done right. **Xmessage** enables you to send a message prompt to an x display on the same system, as long as you know the display you want to send it to. Try:

```
$ DISPLAY=:0 xmessage -center "Hello World!"
```

>_093 Rename files with cp

This is an extension of the way you can use **cp** – **cp** will copy a file and name it to whatever you want, so you can either copy it to another directory and give it a different name, or you can just copy it into the same folder with a different name and delete the original. This also works for renaming with **mv**.

>_094 Manual screenshots

This is something we have to do during Raspberry Pi tutorials, but it works everywhere else. For GNOME-based desktops you can do it in the terminal by calling **gnome-screenshot**, in XFCE it's **xfce-screenshooter**, in LXDE it's **scrot** and so on. This will immediately take a screenshot of what you can see.

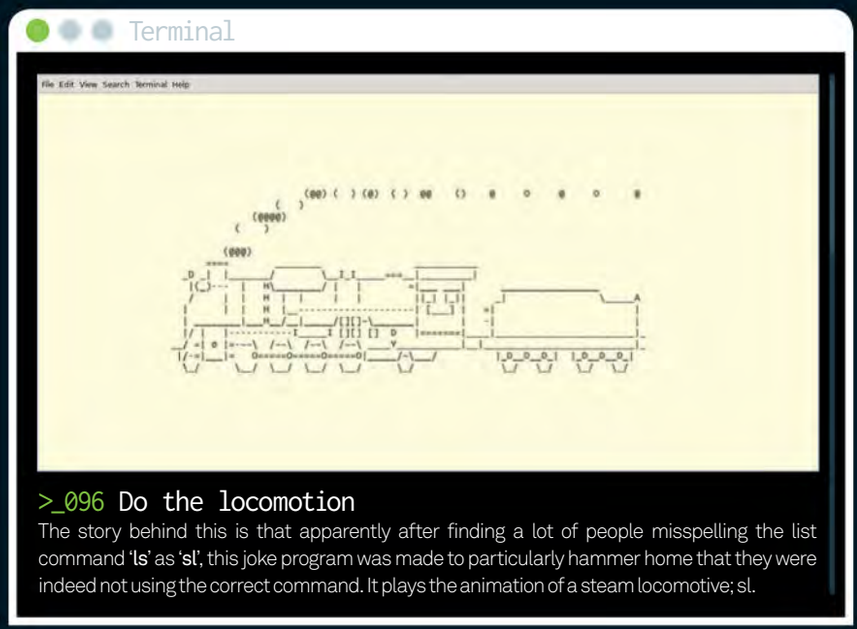
>_095 Delayed screenshots

With the functions from above, you can add a delay to set up a screenshot. This is useful if you want to show the contents of a drop-down menu, or need to pose a shot in general. All of the screenshot tools allow you to delay by adding the **-d** option and then a number of seconds, eg:

```
$ scrot -d 5
```

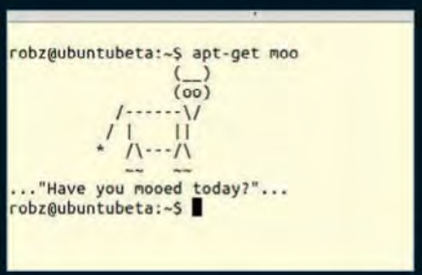
TERMINAL JOKES

>_Easter eggs, gags and other attempts by patterless software engineers to be funny



>_096 Do the locomotion

The story behind this is that apparently after finding a lot of people misspelling the list command '**ls**' as '**sl**', this joke program was made to particularly hammer home that they were indeed not using the correct command. It plays the animation of a steam locomotive; sl.



>_097 What does the cow say

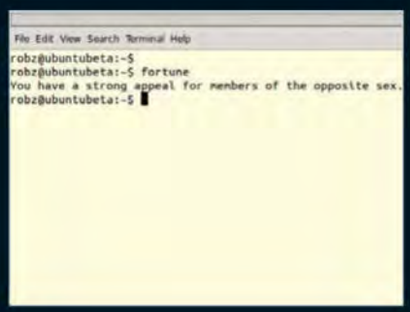
A Debian trick as part of apt-get, **moo** is an easter egg wherein you type **apt-get moo** and a cow asks you if you have mooed today. If you have, in fact, not mooed today then you should immediately find a quiet room and let out a single, loud moo to appease the terminal cow.

>_098 Let it snow

This one uses a Bash script to simulate snow in the terminal. We have added the file you will need to use to **FileSilo.co.uk**, so download it, **cd** to the location of the file and then run it. It's completely pointless, especially in the middle of summer, but it can be relaxing.

>_099 What's in a date

You can actually write **date** into the terminal and it will print out the actual date according to its own internal clock for you, which is very useful. What you can also do is use **ddate** to create a 'discordian date' that prints out an almost dystopian date system.



>_100 Crystal terminal

Our last tip is **fortune**. Call it to get a hint at what your day might involve. Hopefully it's something inspirational to get you started. Remember though, it's just a random output from a string of code. Enjoy!

Start programming in C#

Suggesting the study of C# to a true Unix head is very sound career advice indeed

Resources

MonoDevelop
monodevelop.com

One of the design goals of Microsoft's .NET framework was the creation of a truly portable runtime environment. This aim was furthered by the release of the specification for the intermediary language, a decision that has led developers to suffer under the pain of commercially available decompilers from day one. Redmond's interest in openness even went so far as to rename the intermediary language from Microsoft Intermediate Language to Common Intermediate Language, although these steps were not purely altruistic. Microsoft sought to motivate third parties to create runtimes for their operating system – a project that succeeded largely due to the work of the Mono team.

C#, however, is not of primarily historic interest. It is one of the most commonly used 'traditional' programming languages; its compilates can be run on Windows, OS X, Linux, Android and iOS via a variety of first- and third-party runtime environments.

In addition to that, both Unity and the XNA/MonoGame frameworks have adopted C# as programming language. Game developers working with one of the aforementioned engines can use C# to create the logic that binds the components of their next smash hit.

C# differs from classic scripting languages due to its solid object-oriented base. Its syntax should already be familiar to developers working on C++ and Java.

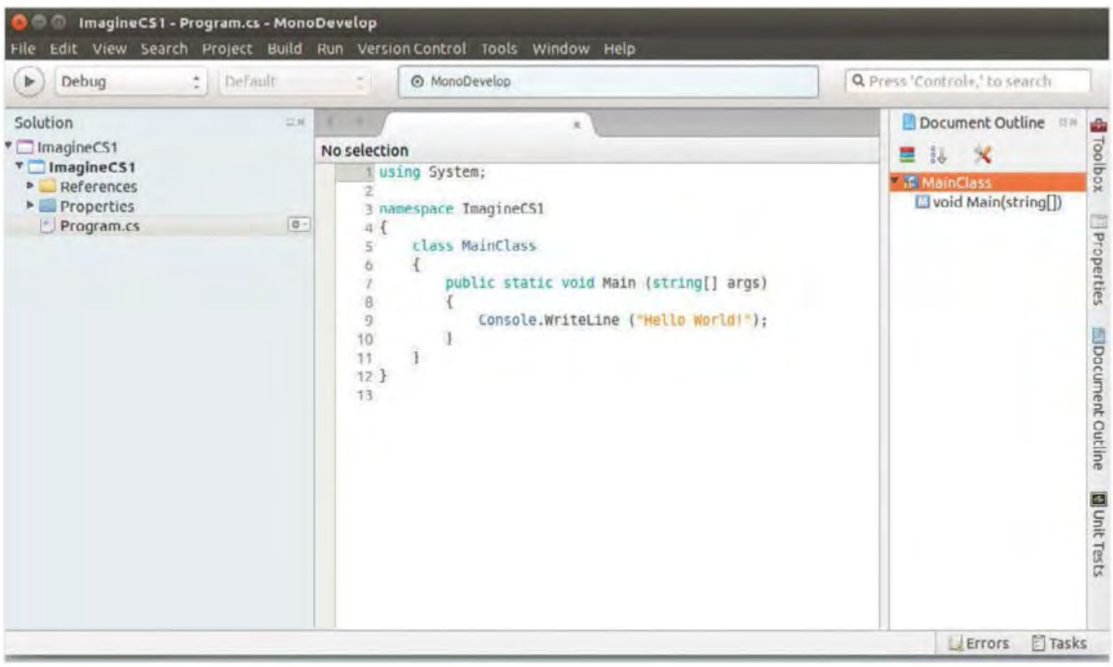
Microsoft's Visual Studio has frequently received all kinds of expansions, striving to make it more competitive with free IDEs, such as Eclipse. Running Windows in a Unix-hosted VM is a possibility due to the widespread availability of high-speed hardware virtualisation. Sadly, developing inside a VM is detrimental for game programmers. Code will exhibit hardware-specific behaviour: if you run in a VM, you are testing the virtualised GPU rather than the one found on the host/target platform.

When working a non-space-constrained system, you should start by downloading the complete VM and the IDE by entering the following commands:

```
sudo apt-get install mono-complete
sudo apt-get install monodevelop monodevelop-nunit
monodevelop-database monodevelop-versioncontrol
```

Hello, MonoDevelop

Even though Ubuntu does provide prepackaged distribution files, the IDE does not integrate itself into the Unity desktop. Firing up MonoDevelop can instead be achieved by opening a terminal, where the command **monodevelop** is to be entered. Next, click the New label in the Solutions field in order to start the creation of a new project skeleton. .NET languages can be combined with a variety of GUI stacks; the IDE will display



Right Type the command **monodevelop** into the terminal to open up the IDE

a selector offering various project types. For now, a C# Console Project shall suffice. Name your project to see the structure shown in the image on page 88.

Developers coming to the world of .NET must acquaint themselves to a new structuring paradigm. A product – be it an aerial torpedo or a game – is stored in a container structure called a Solution. Individual deliverables, such as libraries, resource projects or executables, are then stored as projects. In most development environments, only one project can be active at any given time: it is the one affected by debug and run operations. Selecting the currently active project is done by left-clicking it in the Solution window. The context menu will contain a toggle labelled 'Set as StartUp project'.

It's time to take a look at the code created by the assistant:

```
using System;

namespace ImagineCS1
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            Console.WriteLine ("Hello World!");
        }
    }
}
```

C# code is subdivided into namespaces. Each namespace contains classes, interfaces and similar objects. In fact, the entire system API is exposed via a set of namespaces.

Accessing a member of a namespace is done via its qualified name. So the Console object could be addressed:

```
public static void Main (string[] args)
{
    System.Console.WriteLine ("Hello World!");
}
```

Since namespaces can be nested at an arbitrary depth, programs tend to become unwieldy quickly. This can be solved via the **using** directive, which exposes the public members of the namespace, as shown in the image.

The rest of the project skeleton is not particularly impressive. **Main()** acts as the entry point to Console. WriteLine writes the string passed to it to the command line.

How C is it?

Figuring out how close, or how distant, C and C# are to one another is best accomplished by looking at the type system. C# is a statically typed language: this means that every variable must be of a specified type. The table above-right shows the basic types found in the runtime. C++ programmers tend to get tripped up most often by the array syntax.

Fortunately, this can be solved fairly simply by looking at the following demo project:

```
class MainClass
{
    static int[] myField;
```

C# considers "array" to be a data type. This is a significant improvement in that it permits developers to write **int[] myField**, **myField2**; and get two arrays – in vanilla C, developers trip up. **Main()** demonstrates here the allocation, filling and traversing of the array:

```
public static void Main (string[] args)
{
    myField = new int[100];
    for (int i=0; i<100; i++) {
        myField [i] = i;
    }

    foreach (int x in myField) {
        Console.WriteLine (x);
    }
}
```

The language design of C# placed significant emphasis on the management of collection classes: programmers are provided with turnkey-ready generic collections. Traversing the content is simplified by the **foreach** structure: this permits the declaration of a local variable, which is filled with one element after another.

Microsoft's 'pruning' of the C family also extends to the **switch** statement. Fall-through clauses are not permitted, therefore writing code like this leads to a compiler error:

Short Name	.NET Class	Type	Width	Range (bits)
byte	Byte	Unsigned integer	8	0 to 255
sbyte	SByte	Signed integer	8	-128 to 127
int	Int32	Signed integer	32	-2,147,483,648 to 2,147,483,647
uint	UInt32	Unsigned integer	32	0 to 4294967295
short	Int16	Signed integer	16	-32,768 to 32,767
ushort	UInt16	Unsigned integer	16	0 to 65535
long	Int64	Signed integer	64	-9223372036854775808 to 9223372036854775807
ulong	UInt64	Unsigned integer	64	0 to 18446744073709551615
float	Single	Single-precision floating point type	32	-3.402823e38 to 3.402823e38
double	Double	Double-precision floating point type	64	-1.79769313486232e308 to 1.79769313486232e308
char	Char	A single Unicode character	16	Unicode symbols used in text
bool	Boolean	Logical Boolean type	8	True or false
object	Object	Base type of all other types	-	-
string	String	A sequence of characters	-	-
decimal	Decimal	Precise fractional or integral type that can represent decimal numbers with up to 29 significant digits	128	±1.0 × 10e–28 to ±7.9 × 10e28

Speedy

C# structures can mirror many of the behaviours found in classes: they can contain properties, methods and even private fields. The main restriction is that they can not inherit from one another. Microbenchmarks have shown speed increases of up to 20x over class-based solutions. This is due to them being value types, which can be allocated more easily at runtime. Note that C# tends to be very fast – struct deployment only makes sense in the tightest of loops.

Sealed

By principle, a virtual method can be overridden all the way to China; the **sealed** property informs the compiler that 'the buck stops here'. In the code below, classes derived from B are forced to make do with the implementation of F found in its parent:

```
class A
{
   public virtual
void F() {
      Console.
WriteLine("A.F");
   }
}
class B: A
{
   sealed override
public void F() {
      Console.
WriteLine("B.F");
   }
}
```

```
switch (myField [1]) {
case 1:
    Console.Write ("Found a one");
case 2:
    Console.WriteLine ("One or two");
    break;
}
```

However, the following special case is permitted as the intent to collate multiple cases into one common payload is clearly visible:

```
switch (myField [1]) {
        case 1:
        case 2:
            Console.WriteLine ("One or two");
            break;
        }
```

Show some class

MonoDevelop's skeleton is properly structured in that the main method is located in a class of its own. So far, all of our code is made up of static elements that are shared among all instances.

It is now time to add a new class to our project. Right-click the project and select Add>New File. Use the template General>Empty Class and name it ImagineClass.

Then, modify its code like this:

```
namespace ImagineCS1
{
    public class ImagineClass
    {
        public int myX;
        public ImagineClass (int _x)
        {
            myX = _x;
        }
```

Our class is declared as **public**, meaning it can be accessed from code outside of its containing namespace. Its member variable and its constructor are also public: note the special syntax used to designate the function as constructor method.

C#, of course, is not just limited to public functions. It contains a total of five access modifiers, the following three of which are important:

```
public void doSomething()
{
    doSomethingElse();
    doProtected ();
}
private void doSomethingElse()
{
    Console.WriteLine ("Hello World from
                        ImagineClass" + myX);
}
protected void doProtected()
{
    Console.WriteLine ("Protected World from
                        ImagineClass" + myX);
}
```

Declaring an element as **private** means that it can only be accessed from inside the class. **DoSomethingElse** can only be invoked from within ImagineClass – **doSomething()** displays a valid approach.

Since C# permits the creation of object hierarchies, developers might want to permit derived classes to access the internal state of the original object. This can be accomplished via the **protected** access modifier.

In the following step, change the **Main()** function in order to create a new instance of the class. This is interesting only insofar as that the constructor's parameters are passed in, just as in Java:

```
public static void Main (string[] args)
{
    ImagineClass myCS = new ImagineClass (22);
    myCS.doSomething ();
}
```

C#'s individual way of handling function calls deserves further attention. Take a look at the following snippet of code written out – **modifier()** here receives an integer and an instance of our ImagineClass:

```
static void modifier (ImagineClass myCS, int myInt)
{
    myCS.myX++;
    myInt++;
}

public static void Main (string[] args)
{
    int myInt = 22;
    ImagineClass myCS = new ImagineClass (22);
    Console.WriteLine ("Before " + myInt + " "
                       + myCS.myX);
    modifier(myCS, myInt);
    Console.WriteLine ("After " + myInt + " "
                       + myCS.myX);
}
```

Running this program will yield the output that is shown in the image on the following page. Methods can be permitted to modify the ByVal values passed to them. This is accomplished by adding the **ref** keyword to the declaration of the parameter list and the invocation:

```
static void modifier (ImagineClass myCS,
ref int myInt)
{
    myCS.myX++;
    myInt++;
}

public static void Main (string[] args)
{
. . .
    modifier(myCS, ref myInt);

}
```

Delegate

If we were to name one characteristic that C# features, it most definitely is **delegate**. It can best be described as a hybrid between interfaces (which, incidentally, are available in C#) and function pointers.

Its power can best be demonstrated by looking at the example shown below, which starts out with the declaration of a new delegate:

```
delegate void DelLevitate(String _
whereTo);
```

Both C and Java programmers will wonder about the meaning of this line. It creates a delegate called DelLevitate, which will return void and expects one parameter of the type string. Since the more always leads to the merrier, let us create two functions that fulfil the contract set up in DelLevitate:

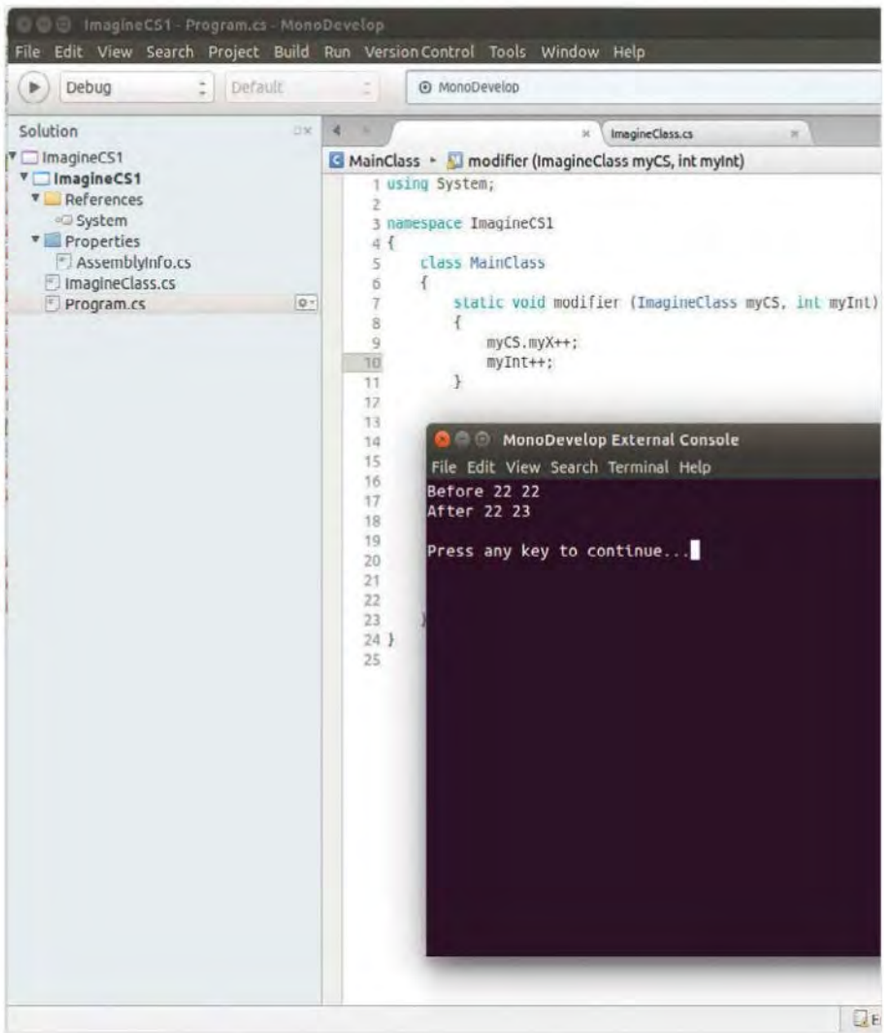
```
public static void saudik(String _
target)
{
    Console.WriteLine (_
target + " is ugly. I don't

want to levitate there!");
}

public static void sylvain(String _
target)
{
    Console.WriteLine ("Of
course! Levitating to "
                        + _target);
}
```

With that, it is time to create a new instance of the delegate. Be aware that the name of the function used in the declaration can now be used as data type.

In the next step, the delegate will be created by passing in a function pointer. Adding a second one then transforms it into a multicast delegate: this is a special form that relays incoming information to all functions that are inscribed within. Finally, the delegate is invoked with a string as follows:

```
public static void Main (string[] args)
{
    DelLevitate mySaudik;
    mySaudik = sylvain;
    mySaudik += saudik;
    mySaudik ("Town");
}
```

Now run the program in order to treat yourself to the display of both messages.



Above **Modifier()** receives an integer and an instance of ImagineClass

> Fortunately, high similarity to C++ and Java means that developers can 'grok' the language as they code

Pointer and unsafe

By default, attempts to create pointer data types are blocked by the compiler. Using them requires the use of the **unsafe** keyword, which informs the runtime about the 'peculiarities' of the code contained within it.

Sadly, harnessing the power of unsafe is not particularly sensible as many runtimes, especially those on Windows Phones, do not permit the execution of sections marked unsafe.

Next step

Books describing C# in its entirety tend to contain hundreds of pages. The highlights that have been shown here cover only the most important features.

Fortunately, high similarity to C++ and Java means that developers can 'grok' the language as they code – and as ever, if something remains ambiguous, auntie Google tends to answer nice and quickly. ■

Use your Kinect with Linux

Microsoft's Kinect sensor provides a surprisingly convenient way to collect depth and colour information

Resources

Qt Creator
bit.ly/1zhD2ny

Some years ago, Microsoft's Kinect was all the rage. Since then the sands of time have whisked the demo booths and their attendants away, and the second generation sensor has hit the market. A few years after the initial release, it is now time to take a look at what the Kinect can actually do in practice.

Hard-core Linux fans will be delighted to hear that the Kinect was not a development by Microsoft proper: instead, the sensor was based on technology developed by an Israeli start-up called PrimeSense. Microsoft changed the way the hardware worked when starting to design the Kinect – a modification of significant importance to coders. PrimeSense used an embedded processor on the sensor, thereby returning skeletal, facial, depth and colour frames to the workstation. Microsoft's system opted to do the skeletal tracking on the workstation and the Xbox was fast enough to use.

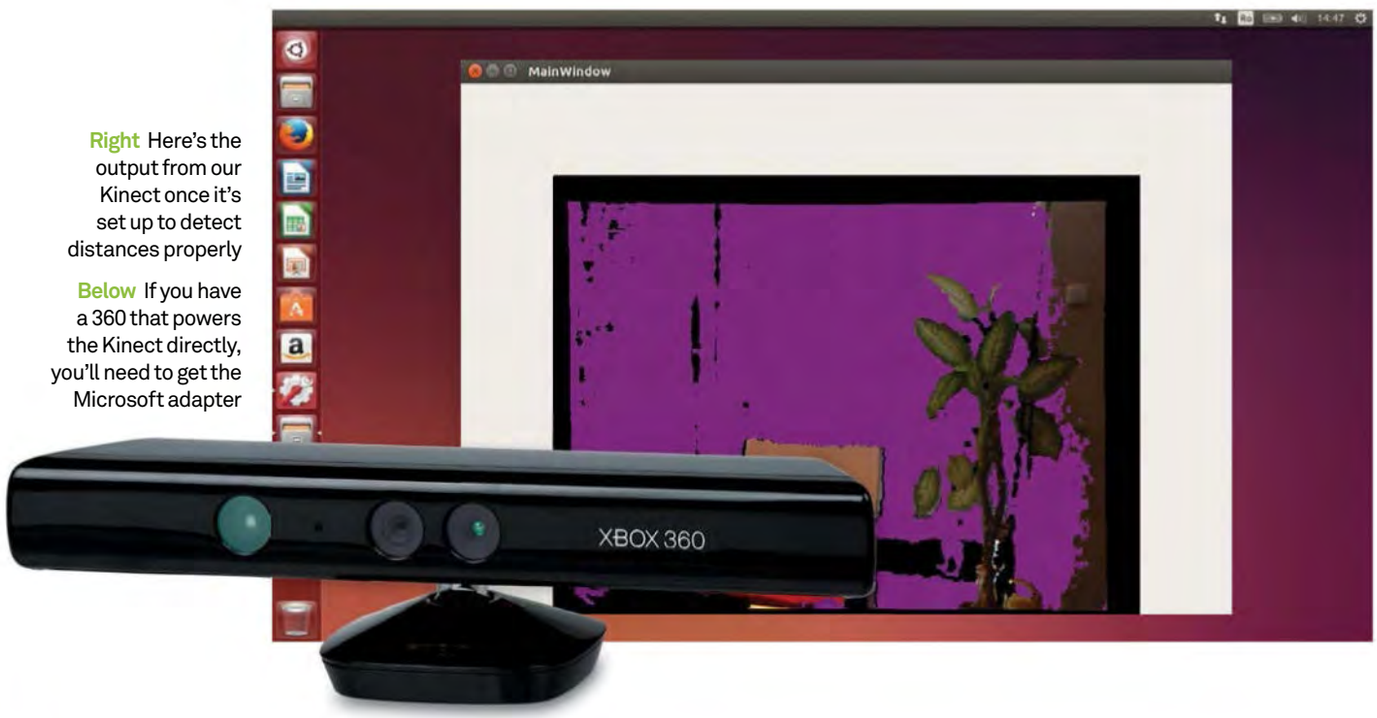
At the time of writing two types of Kinect exist. We will focus on the widespread original Kinect, mentioning its uncommon and finicky successor model only in select places.

When purchasing a Kinect, multiple versions are available. The simplest to use is the Kinect for Windows. It is licensed for desktop use, comes with a power supply and supports the Near Mode, which we won't cover here. Kinects intended for the Xbox 360 can also be used, though they will require an adapter to transform its power plug into an accessible format.

Installation

Kinect drivers for UNIX are developed in the freenect project. When working on older versions of Ubuntu, a custom PPA needs to be used in order to get a recent version of the library. This is important as a binary break took place around the February 2011 version. Our example code uses the second version of the API and cannot be compiled with the first one:

```
tamhan@TAMHAN14:~$ sudo add-apt-repository ppa:floe/
libtisch
[sudo] password for tamhan:
libTISCH is a multitouch development platform, based
on OpenGL.
For more information, see the homepage at http://
tisch.sf.net/.

More info: https://launchpad.net/~floe/+archive/
ubuntu/libtisch.
Press [ENTER] to continue or Ctrl+C to cancel
adding it:
. . .
tamhan@TAMHAN14:~$ sudo apt-get update
tamhan@TAMHAN14:~$ sudo apt-get install libfreenect
libfreenect-dev libfreenect-demos
```



Right Here's the output from our Kinect once it's set up to detect distances properly

Below If you have a 360 that powers the Kinect directly, you'll need to get the Microsoft adapter

It is possible to enable User-level access by modifying the file /etc/udev/rules.d/66-kinect.rules as follows:

```
#Rules for Kinect ###################
############
SYSFS{idVendor}==“045e”,
SYSFS{idProduct}==“02ae”,
MODE=“0660”,GROUP=“video”
SYSFS{idVendor}==“045e”,
SYSFS{idProduct}==“02ad”,
MODE=“0660”,GROUP=“video”
SYSFS{idVendor}==“045e”,
SYSFS{idProduct}==“02b0”,
MODE=“0660”,GROUP=“video”
### END ##############################
###########
```

Finally, add yourself to the video group in order to be able to access the sensor:

```
$ sudo usermod -a -G video YOURUSERNAME
```

When working with Ubuntu14.04, installation is somewhat easier. Just enter the following command into your terminal:

```
sudo apt-get install freenect
```

Running a first test is easy. Connect the sensor to your power supply and workstation, then enter the following command:

```
tamhan@TAMHAN14:~$ freenect-glview
Kinect camera test
Number of devices found: 1
GL thread
write_register: 0x0105 <= 0x00
```

On some machines, a kernel module will block direct access to the sensor. If this is the case, enter the following commands in order to efficiently banish it:

```
sudo modprobe -r gspca_kinect
sudo modprobe -r gspca_main
```

Importing freenect

With that, it's time to start up the trusty Qt Creator. Spawn a new project of the type QtGui. Its profile must be extended by the following segment, which imports the freenect library and forces the compiler to work in x86 mode:

```
QT       += core gui
greaterThan(QT_MAJOR_VERSION, 4): QT +=
widgets
TARGET = ImagineKinect
TEMPLATE = app
CONFIG +=i386
DEFINES += USE_FREENECT
LIBS+=-lfreenect
```

Freenect provides a simple C interface which must be implemented by hand to start our sensor. We will enclose the logic in a class called KinectHost. It must be derived from QThread and has the following structure:

```
class KinectHost:public QThread
{
    Q_OBJECT
public:
    KinectHost(QObject* parent);
    ~KinectHost();
    void run();
    void internalizeDepth(void* depth);
    void internalizeColor(void *video);
signals:
    void colorHere(void* data);
    void depthHere(void* data);
public:
    QMutex* myMutex;
    static KinectHost* mySelfRef;
    freenect_context *myContext;
    freenect_device *myDevice;
    unsigned char* myColorBuffer;
    unsigned short int* myDepthBuffer;
    bool myDepthTaken;
    bool myColorTaken;
};
```

Here we declare a set of signals that are to be emitted as frames arrive from the sensor. Secondarily, we provide two struct pointers that act as data storage classes for the Freenect library. Finally, two flags named 'myDepthTaken' and 'myColorTaken' are implemented. We will look at their role in a few seconds.

Working with freenect

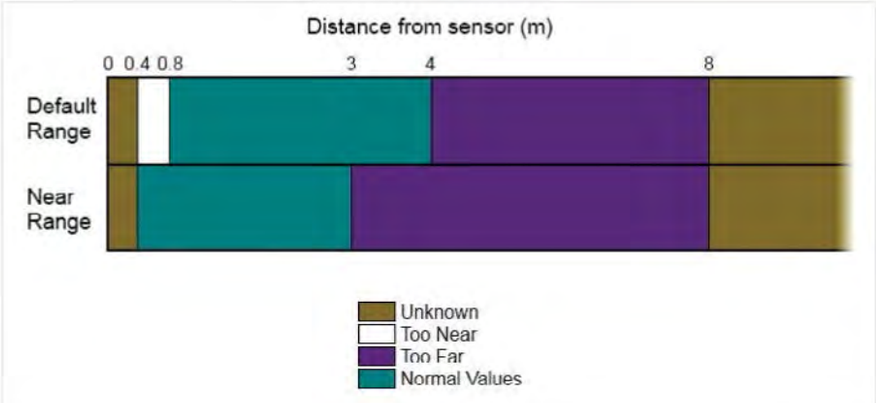
Initialising freenect is a relatively involved procedure with similarities to classic imaging toolchains. We start out by invoking freenect_init to allocate memory. Next, setLogLevel is invoked – it specifies the amount of information emitted to the standard output. When this is done, Open Device is used to activate the first Kinect connected to the machine:

```
KinectHost::KinectHost(QObject* parent):
  QThread(parent)
```

Scanning distance

Objects need to be kept at a specific distance to the sensor to be scanned accurately (see above diagram). On the Kinect for Windows, a dedicated Near Mode is also available that attempts to reduce the minimal scanning distance using interpolation, but the reachable gain is limited by the similar hardware used.

```
{
  myMutex=new QMutex();
  mySelfRef=this;
  myDepthTaken=myColorTaken=false;
  if(freenect_init(&myContext,NULL)<0)
  {
    qDebug() << “Could not initialize
library”;
  }
  freenect_set_log_level(myContext,FREENECT_
LOG_WARNING)
  if(freenect_num_devices(myContext)<=0)
  {
   qDebug() << “No Kinect”;
   freenect_shutdown(myContext);
  }
  if(freenect_open_device(myContext,&my
evice,0)<0)
  {
    qDebug() << “Kinect 0 blocked”;
    freenect_shutdown(myContext);
  }
  myColorBuffer=(unsigned char*
malloc(640*480*3);
  freenect_set_video_callback(myDevice,
```



Second generation

Microsoft showcased the incredible scanning prowess of the second-gen Kinect in a variety of scenarios. Sadly, its high resolution cameras ensure a figety and complex deployment: parsing the huge data streams requires the use of GPU acceleration and loads of USB 3.0 bandwidth. Linux developers are unlikely to profit from this pricey version.

```
videoCB);
  freenect_set_video_buffer(myDevice, myColorBuffer);
  freenect_frame_mode myMode=freenect_find_video_
mode(FREENECT_RESOLUTION_MEDIUM, FREENECT_VIDEO_RGB);
  freenect_set_video_mode(myDevice,myMode);
  freenect_start_video(myDevice);
```

The aforementioned binary break lurks in the setup of the image processing toolchain. The first generation of the library permitted you to specify the format of the stream directly. In versions released after March 2011, the video frame format must be declared in two steps: find mode returns a list of modes based on criteria, and setVideoMode uses it for actual setup. The initialisation of the depth stream is done along the same scheme – the only difference involves the passing of freenect DepthRegistered. This constant specifies that the sensor is to return correlated values scaled in millimetres:

```
myDepthBuffer=(unsigned short int*)malloc(640*480*2);
freenect_set_depth_callback(myDevice, depthCB);
freenect_set_depth_buffer(myDevice, myDepthBuffer);
myMode=freenect_find_depth_mode(FREENECT_
RESOLUTION_MEDIUM, FREENECT_DEPTH_REGISTERED);
freenect_set_depth_mode(myDevice,myMode);
freenect_start_depth(myDevice);
}
```

Being a C library, freenect cannot provide itself with CPU power. We solve this problem by regularly invoking freenect ProcessEvent from the QThread's run method:

```
void KinectHost::run()
{
  while(1==1)
  {
    if(freenect_process_events(myContext)<0)
    {//Provide CPU power
      qDebug() << "Cant process events";
    }
  }
}
```

Using the frames

With that, it's time to handle the frames provided by the sensor. This has to be done in two classic C functions:

```
static inline void depthCB(freenect_device *dev,
void *depth, uint32_t timestamp)
{
  KinectHost::mySelfRef->internalizeDepth(depth);
}
static inline void videoCB(freenect_device *dev,
void *video, uint32_t timestamp)
{
  KinectHost::mySelfRef->internalizeColor(video);
}
```

We use the static mySelfRef field in order to invoke the internalise methods stored in the class itself. Since their code is similar, we will print only the depth handler:

```
void KinectHost::internalizeDepth(void* depth)
{
  QMutexLocker aLocker(myMutex);
  if(myDepthTaken==true)
  {
    unsigned char* mySecondBuffer=(unsigned
char*)malloc(640*480*2);
    memcpy(mySecondBuffer, depth, 640*480*2);
    emit depthHere(mySecondBuffer);
    myDepthTaken=false;
  }
```

Our code starts out by locking the access Mutex. It then checks whether the last frame was collected. If this is the case, we proceed to allocating a new buffer of memory and copy the data provided from the work area into the newly allocated buffer. Finally, the signals above are emitted. Their parameter is a pointer to the newly allocated buffer. Purists might wonder about the reasons for this copying; Freenect works with two internal buffers which get reused as frames come in. We must copy the payload data out of the work area as fast as possible in order to avoid race conditions.

Setting up the display

With that out of the way, let's proceed to the implementation of the QMainWindow responsible for displaying data. Its constructor looks like this:

```
MainWindow::MainWindow(QWidget *parent) :
  QMainWindow(parent),
  ui(new Ui::MainWindow)
{
  ui->setupUi(this);
  myHost=new KinectHost(this);
  connect(myHost, SIGNAL(colorHere(void*)),this,
SLOT(colorHere(void*)));
  connect(myHost, SIGNAL(depthHere(void*)),this,
SLOT(depthHere(void*)));
  myHost->start();
  myHost->myColorTaken=true;
  myHost->myDepthTaken=true;
  colorFlag=depthFlag=false;}
```

We start out by creating a new instance of the Kinect host class. Its signals are connected to the slots in the form. The thread is then kicked off by a call to start – the freenect library needs CPU time for handling the frames. Finally, both flags are set to true in order to make the host provide data. Handling colour is easier than handling depth. Due to this, we will start out by looking at the event handler responsible for incoming camera frames:

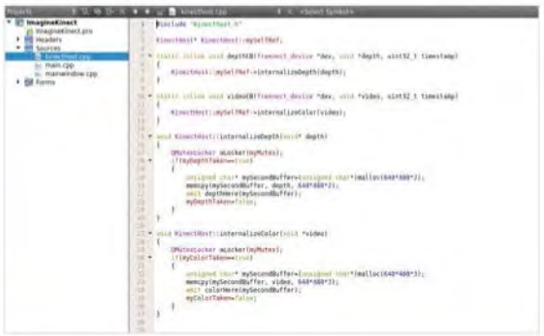
```
void MainWindow::colorHere(void* _data)
{
```

```
  qDebug("Color here");
  if(myColorImage!=NULL)delete myColorImage;
  myColorImage=new QImage(640,480, QImage::Format_
RGB32);
  unsigned r, g, b;
  u_int8_t* myData=(uint8_t*) _data;
  for(int x=1;x<640;x++)
  {
    for(int y=0;y<480;y++)
    {
      r=myData[3*(x+y*640)+0];
      g=myData[3*(x+y*640)+1];
      b=myData[3*(x+y*640)+2];
      myColorImage->setPixel(x,y,qRgb(r,g,b));
    }
  }
  free(_data);
  colorFlag=true;
  myHost->myColorTaken=true;
}
```

The Kinect provides frames which are an array of unsigned short integers. Our loop must parse the data to find the individual pixel colours, which are then written into a QImage. When working with practical code, it might be faster to use memCpy in order to save processing time.

Finally, the buffer allocated in KinectHost must be released by invoking free. Not doing so creates a memory leak significant enough to bring down high-end machines in just seconds. Finally myHost->myColorTaken is set to true in order to announce that the colour frame was destroyed and that a new one can be provided. Handling depth data is a bit more complex, as shown below:

```
void MainWindow::depthHere(void* _data)
{
  if(colorFlag==true)
  {
    colorFlag=false;
      u_int16_t* myData=(uint16_t*) _data;
      if(myCombinedImage!=NULL)delete
myCombinedImage;
      myCombinedImage=new QImage(640,480,
QImage::Format_RGB32);
      for(int x=1;x<640;x++)
    {
      for(int y=0;y<480;y++)
      {
        int calcval=myData[(x+y*640)];
        if(calcval!=FREENECT_DEPTH_MM_NO_VALUE)
        {
          if(calcval>1500 && calcval<3000)
          {
            myCombinedImage->setPixel(x,y,
myColorImage->pixel(x,y));
          }
          else
```



```
          {
            myCombinedImage-
>setPixel(x,y,qRgb(128,0,128));
          }
        }
      }
    }
    free(_data);
    myHost->myDepthTaken=true;
    repaint();
  }}
```

By and large, the depth frames are not too different from the colour ones. They are made up of unsigned 16-bit integers, representing the distance of the element from the sensor's main plane. We determine this and then use the returned value to decide the colour of the image that is to be drawn. Our application will only draw objects at a distance between 1,500 and 3,000 millimetres. All other elements will appear in either black or pink. Redrawing elements is permitted only in response to a paint event. Our depth event handler enqueues one by calling redraw – its handling code simply writes the contents of the bitmap to the screen as follows:

```
void MainWindow::paintEvent(QPaintEvent *_p)
{
  qDebug("Want to render");
  if(myCombinedImage!=NULL)
  {
    QPainter p(this);
    p.drawImage(ui->widget->pos().x(),ui->widget-
>pos().x(),*myCombinedImage);
  }
}
```

Final thoughts

Our distance-discriminating webcam is now ready. Run it to see a small part of your surroundings. When compared to Microsoft's examples, our app isn't very impressive due to the design deficiency. Skeletal correlation is handled on the host, as Microsoft is not willing to provide the algorithms used in its Windows SDK. PrimeSense used to offer a tracking library of its own, but it was taken offline after Apple acquired the firm. Due to its insecure future potential, we won't discuss OpenNI.

Left For more advice on setting up your Kinect, check out **openkinect.org**

Intermediate AWK programming

Take your AWK knowledge to the next level with this comprehensive step-by-step tutorial

Resources

Text editor

Gawk
gnu.org/software/gawk

Mawk
invisible-island.net/mawk

We will begin this tutorial with a touch of historical information about AWK. The rest of the tutorial will deal with more advanced AWK programming techniques such as the use and development of functions, BEGIN and END code blocks, etc. But first, a little background on what we're working with.

The name AWK originated from the names of its three authors: Alfred Aho, Peter Weinberger and Brian Kernighan. It is a very handy pattern-matching programming language that is Turing-complete and was developed at the famous AT&T Bell labs. The biggest difference between the various AWK variants and the original version of AWK is that newer versions support a larger set of built-in functions and variables, where the original did not.

Despite its many capabilities, AWK cannot solve every problem equally well. It is more suited for situations in which you have to process large amounts of text data, such as log files, LaTeX files, source code files, and so on. AWK enables you to analyse, extract and report data, making it a very useful framework. It also supports arrays, associative arrays and regular expressions.

01 Installing AWK

Although AWK comes with every Linux distribution, this article is going to use a specific version called gawk. You can find out if gawk is already installed on your Linux by using:

```
$ which gawk
```

If it is not installed, you can install gawk on a Debian system by running the **apt-get install gawk** command as root. Next, you can find your version of gawk by running **gawk –V**. All AWK code will be shown in the form of autonomous executable programs in the following format:

```
#!  /usr/bin/gawk -f
...
AWK code
...
```

You can give executable permissions to an AWK script with:

```
$ chmod 755 filename.awk
```

```
code$ # Without the nextfile statement
code$ # FATAL ERROR
code$ ./BEGINFILE.awk *
New file anagrams.awk
New file BEGIN.awk
New file BEGINFILE.awk
New file CANNOTREAD
gawk: ./BEGINFILE.awk:8: fatal: cannot open file `CANNOTREAD' for reading (Permission
 denied)
code$ # With the nextfile statement
code$ cat BEGINFILE.awk
#! /usr/bin/gawk -f

BEGIN {

}

BEGINFILE {
    print "New file", FILENAME

    # Check if there is an error while trying to read the file
```

Right Using the nextfile function with the ERRNO variable helps you avoid fatal errors when processing multiple data files

02 How AWK works

AWK reads its input line by line and accordingly splits each line into fields. AWK has many built-in variables that are automatically initialised and it supports user-defined variables. AWK also has built-in functions for string, number, time and date manipulation.

The **NF** variable holds the number of fields in the current input record; each record can have a different number of fields. The **FS** variable is the input field separator, and its default value is the single space that matches any sequence of spaces and tabs in a single separator. Additionally, any number of leading or trailing spaces and tabs is ignored. If its value is the null string, then each character in the current line becomes a separate field.

```
mtsouk@mail:~/docs/article/working/intAWK.LUD/code$ ./BEGIN.awk BEGIN.awk BEGIN.awk
Read 6 records in total!
mtsouk@mail:~/docs/article/working/intAWK.LUD/code$ ./BEGIN.awk BEGIN.awk
Read 3 records in total!
mtsouk@mail:~/docs/article/working/intAWK.LUD/code$ cat BEGIN.awk
#! /usr/bin/gawk -f

END { print "Read", NR, "records in total!" }
mtsouk@mail:~/docs/article/working/intAWK.LUD/code$
```

03 Using BEGIN and END

AWK has two patterns named **BEGIN** and **END**, each of with has a special meaning and function. It should be noted that both of them are executed only once: before getting any input and after processing input. They are very useful for performing start-up and clean-up actions in your AWK programs.

Although it is not required to have the BEGIN rule at the beginning of your AWK program and the END rule at the end, it is considered good practice to put them there. An AWK program can have multiple BEGIN and END rules.

If an AWK program has only BEGIN rules without any other code, the program terminates without reading any of the specified input. However, if an AWK program contains only END rules without any additional code, all the specified input is read in case the END rule needs to reference the **FNR** or **NR** variables.

> The first gawk version was introduced in 1986. It is still the most powerful and popular version of AWK

The FNR variable is used for keeping the total number of records that have been read from the current input file only, whereas the NR variable is used for keeping track of the number of records that have been read so far.

04 Using BEGINFILE and ENDFILE

Two other patterns with a special function are **BEGINFILE** and **ENDFILE** – please note that these only work on gawk. BEGINFILE is executed before gawk reads the first record from a file and ENDFILE is executed after gawk is done with the last record of a file.

ENDFILE can be very convenient for recovering from I/O errors during processing; the AWK program can pass the control to ENDFILE (instead of stopping abnormally) and set the **ERRNO** variable that describes the error that has happened. Gawk clears the ERRNO variable before it starts processing the next file. Similarly, the **nextfile** statement, when used inside BEGINFILE, enables gawk to move to the next data file instead of exiting with a fatal error and without executing the ENDFILE block.

The presented code showcases their usage by printing the total number of files read as well as the filename of each item. The code also reports if there is a problem when reading a file. The printing of the filename is done with the help of the **FILENAME** variable.

Versions of AWK

The original version of AWK was programmed in the first UNIX versions back in 1977. The first gawk version was introduced in 1986. Michael Brennan wrote another AWK variant named mawk and Nelson H.F. Beebe programmed pawk. Gawk is still the most powerful and popular version of AWK.

04 Using BEGINFILE and ENDFILE

```
New file BEGINFILE.awk
New file CANNOTREAD
Cannot read CANNOTREAD Processing next file!
New file functions.awk
New file special.awk
New file wc.awk
Total number of files processed:  6
code$ cat BEGINFILE.awk
#! /usr/bin/gawk -f

BEGIN {
    numberOfFiles = 0
}

BEGINFILE {
    print "New file", FILENAME

    # Check if there is an error while trying to read the file
    if (ERRNO )
    {
        print "Cannot read", FILENAME, "Processing next file!"
```

Left If there's an error, a nextfile statement in BEGINFILE or ENDFILE can help AWK proceed

Regular expressions

Regular expressions play a key role in AWK. Regular expressions are enclosed in slashes. In its simplest form (**/text/**), a regular expression matches any string that contains "text". Unless otherwise specified, regular expressions are case sensitive and can be used standalone or in conditional expressions. When used properly, they are very powerful tools, but they can unfortunately also cause nasty bugs.

```
code$ ./special.awk *.awk
ARGV[0] = gawk
ARGV[1] = anagrams.awk
ARGV[2] = BEGIN.awk
ARGV[3] = BEGINFILE.awk
ARGV[4] = functions.awk
ARGV[5] = special.awk
ARGV[6] = wc.awk
code$ cat special.awk
#! /usr/bin/gawk -f

BEGIN {
    for (i = 0; i < ARGC; i++)
        printf "ARGV[%d] = %s\n", i, ARGV[i]
}
code$ ll special.awk
-rwxr-xr-x 1 mtsouk mtsouk 107 Jun 27 20:31 special.awk
code$
```

05 Special filenames

The FILENAME variable dynamically takes its value from the file that is currently being processed. If AWK is reading from standard input, the FILENAME value is set to "-". AWK also supports **ARGC** and **ARGV**; you should be familiar with both variables if you have ever done any C or C++ programming.

Gawk supports the following three special and standard filenames: /dev/stdin, /dev/stdout and /dev/stderr. These can also be accessed using file description notations as /dev/fd/0, /dev/fd/1 and /dev/fd/2, respectively. Closing any of these file descriptors from an AWK program using the **close()** function results in unpredictable behaviour and should be avoided. Closing any other open file descriptor results in the actual closing of the file descriptor.

Gawk also supports network connections acting as either a server or a client using the following notation:

```
/net-type/protocol/local-port/remote-host/
remote-port
```

Talking more about the networking capabilities of gawk is beyond the scope of this article.

```
code$ cat functions.awk
#! /usr/bin/gawk -f

function isnum(x) { return(x==x+0) }

function sumToN(n)
{
    sum = 0
    if (n < 0) { n = -n }
    if ( isnum(n) )
    {
        for (j = 1; j <= n; j++) { sum = sum + j }
    }
    else { return -1 }
    return sum
}

{
    for (i=1; i<=NF; i++)
    {
        print $i, "\t:", sumToN($i)
    }
}
```

06 Functions in AWK

AWK has built-in functions, but also allows you to develop your own. The general format of a user-defined function in AWK is as follows:

```
function function_name(parameter list)
{
various AWK commands
}
```

As you can see, it looks similar to the function implementation of a typical programming language.

The implanted function will calculate the sum of all integers from one up to a given limit. As you can see, the presented AWK code does two checks when processing the input. The first check looks to see if the input is a negative number. If yes, the absolute value of the input will be used as the limit. The second check is for making sure that the input is an integer that does not contain any invalid characters. The **isnum()** function uses the fact that non-numeric strings in AWK are treated as having the value zero when used in mathematical operations.

Please note that you will need to use different variable names for the two **for** loops of the program.

```
os 2
bb 4 ww -3
code$ cat sorting.awk
#! /usr/bin/gawk -f

# read every line into an array
{
  toBeSorted[NR] = $0
}

# This is where the actual sorting happens!
END {
    do
    {
        changed = 0
        for(i=1; i<NR; i++)
        {
            if ( toBeSorted[i] > toBeSorted[i+1] )
            {
                temp = toBeSorted[i]
                toBeSorted[i] = toBeSorted[i+1]
                toBeSorted[i+1] = temp
                changed = 1
            }
        }
    } while (changed == 1)

    # print it
```

07 Sorting in AWK

AWK can also be used for programming and sorting algorithms. Sorting large amounts of data with the help of AWK might not be the fastest implementation, but it works.

The trick is done using the **$0** variable that represents the entire line or record read. You get $0 when reading the input file and create a new array where each element is the whole line. The NR variable is used for creating unique array indices. The new array, called **toBeSorted**, is then being sorted in the END part of the program; in other words, after you are done reading the whole input! As you can also see, the AWK code looks pretty much like C code.

With the help of ENDFILE and END, you can develop a similar program that can sort each individual file as well as the entire input of all files read.

08 Programming wc in AWK

This step will show you how to write an AWK program that simulates the function of the **wc** command line utility. As you can guess, this AWK script will use an END block for presenting the summary results, as well as an ENDFILE block for printing results for each individual file right after it has been processed.

The **printf** function is similar in use to the C printf function. It is exceptionally useful when you want better control of what is printed. The current version of wc.awk does not support command line options such as **wc –l**.

The **moreThanOne** parameter lets the program know if it has to process more than one file. In that case, wc.awk will have to print totals.

```
code$ cat awkprof.out
       # gawk profile, created Mon Jun 29 18:07:04 2015

       # Rule(s)

    5  {
    9          for (i = 1; i <= NF; i++) {
    9                  print $i, "\t:", sumToN($i)
               }
       }


       # Functions, listed alphabetically

    9  function isnum(x)
       {
    9          return (x == x + 0)
       }

    9  function sumToN(n)
       {
    9          sum = 0
    9          if (n < 0) { # 1
    1                  n = -n
```

Left Here you can see how many times a rule or statement was used at the beginning of each line

The main program where the counting is taking place, has the following lines:

```
{
    chars += length($0) + 1
    lines++
    words += NF
}
```

This implementation is by far the smallest and easiest program to understand. Try to program the same utility in C or Python to understand its true complexity.

09 Finding word signatures in AWK

The letters of a word sorted in alphabetical order compose the signature of the word. In order to sort the letters of each word, you will need to use a sorting function – this can be done simply with the **asort()** function that is provided by gawk.

Word signatures help you find word anagrams easily. An anagram uses exactly the same letters from the original word to make a new word.

The presented algorithm splits each word into individual letters before sorting them using asort(). Then, it joins all of the letters before returning the signature of the word.

```
code$ cat signatures.awk
#! /usr/bin/gawk -f

function createSignature(word, a, i, n, result)
{
    n = split(word, a, "")
    asort(a)

    for (i=1; i<=n; i++) { result = result a[i] }
    return result
}

{
    for (j=1; j<=NF; j++)
```

> This AWK script will use an END block for presenting the results, and an ENDFILE block for printing the results

10 Profiling AWK programs

It is possible to profile AWK programs with the help of the **--profile** gawk option. The default filename with the profiling information is **awkprof.out**. You can use the **--profile=anothername.ext** option to change the name. The functions.awk program will be profiled as an example. In order to create profiling information for functions.awk, you must remove the first line and save it as profile.awk. Then, you should execute this next command:

```
$ gawk --profile -f profile.awk function.input
$ ls -l awkprof.out
-rw-r--r-- 1 mtsouk mtsouk 511 Jun 29 18:07
awkprof.out
```

The **–f** parameter tells gawk that it is going to read the AWK code from a file that will be given as a command line argument. The awkprof.out file contains the profiling information. The count on the left of each line shows how many times a rule or statement was used. The second count, to the right, shows how many times the action for the rule was executed. Only appropriate lines have two counts. Functions are listed in alphabetical order.

Please bear in mind that the more times you execute the code, the more accurate the profiling results will be, so giving a bigger input will help immensely.

Launch your first Docker container

Containers are the coolest tech right now – everyone wants to run their software in Docker containers

Resources
Docker
docker.com

Linux containers provide a simple way to run applications in an isolated environment. Applications are bundled with the libraries and other required binaries, and then run in an isolated manner via containers. Containers have been around for some time – projects like LXC, OpenVZ and so on were released around 2005 – but things really changed with Docker. Not only did containerisation go mainstream among users, but more businesses are now using Docker for their production systems.

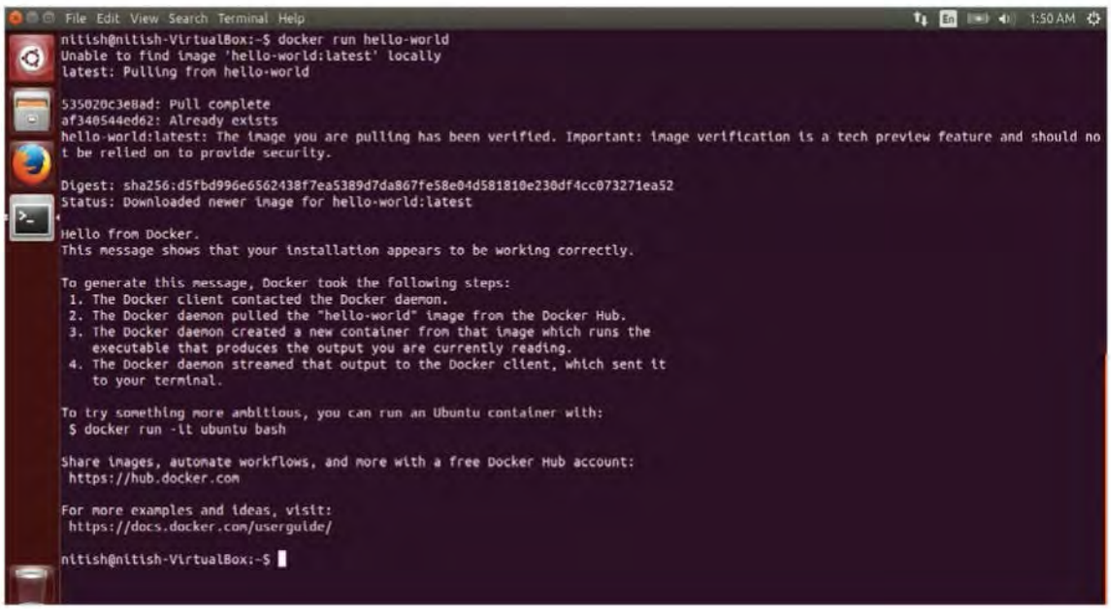
Before we jump onto Docker and its basics, let's try to get some clarity over one of the basic questions – the difference between containerisation and virtualisation. Virtualisation involves creating a target kernel on the host machine, so you will have as many kernels as virtual machines on your system. Now this may be required in some scenarios, but having a separate kernel is generally a big overhead. Containers, on the other hand, just isolate the runtime of the application you need to run, but at the lower layer, all the containers on your machine use the same kernel – this makes them lightweight and very fast to deploy. However, this also means that all of the containers are tied to one OS and you don't have the freedom to run apps for different operating systems on one host. You can install Docker on any OS, though.

01 Docker architecture

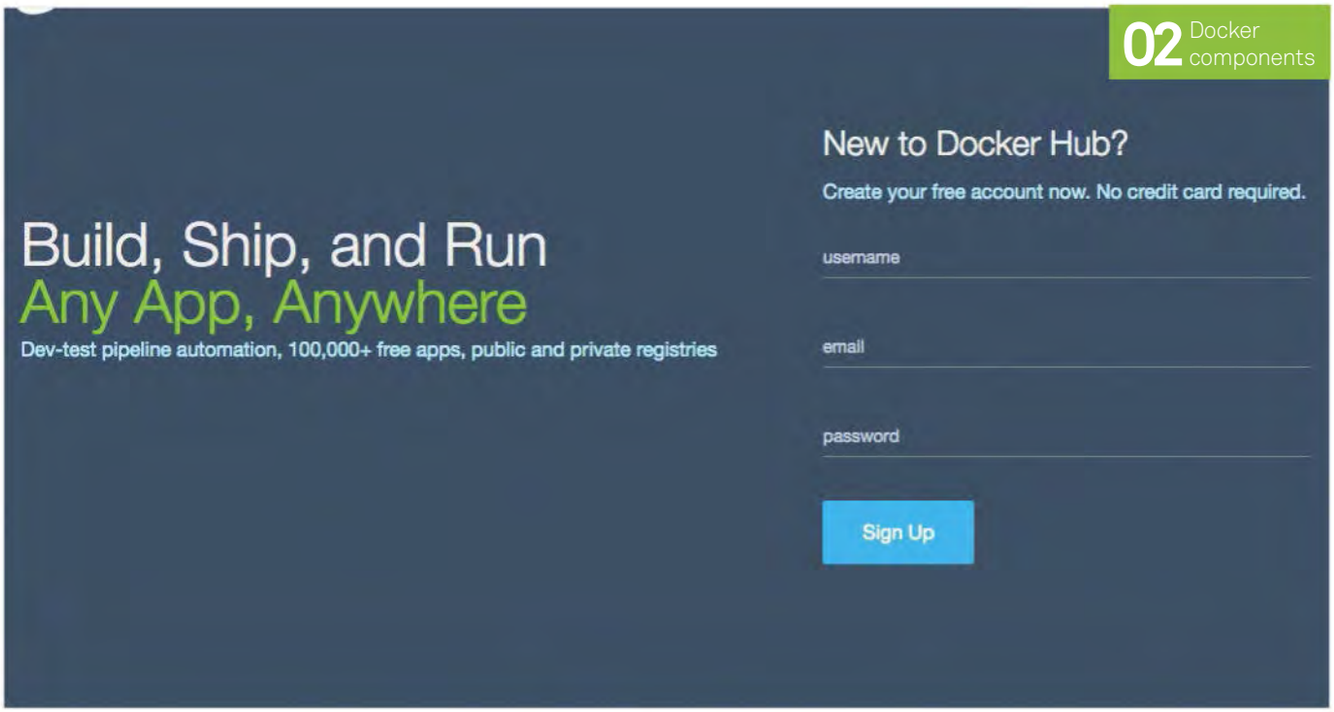
Docker is based on client-server architecture: Docker client accepts the commands from users and then talks to the Docker daemon that does the core tasks such as building, running and even distributing the Docker containers. Note that Docker client and the daemon communicate via RESTful APIs, therefore the communication is asynchronous. Also, this makes the system loosely coupled: it is possible to have the client and daemon run on the same system, or different systems, based on your requirements. However, it is important to understand that the Docker daemon should run on the host machine; for example, the machine where containers are supposed to run and that the client runs on the system being used as the interface, because the Docker client is the primary interface to Docker for the user.

02 Docker components

Apart from the server and the client, there are three other components of Docker that are important to clarify in order to understand its overall working. These are Docker images, containers and registries.



Right A Docker container can be created using command **docker run <image-name>**



A Docker image is a read-only template in a predefined format that the Docker daemon can run – so any application that you want to run as a Docker container first needs to be made as a Docker image. Images generally contain the base operating system, the libraries and binaries required for the application to be run, and obviously the application itself. Docker provides simple ways to build new images or update old ones. You can also download images that others have built via the Docker registry.

A Docker registry is a storage and content delivery system, holding Docker images. You can interact with a registry using the **push** and **pull** commands. A registry can be private or public depending on your requirements. Docker also provides the official Docker registry as a SaaS (software-as-a-service) platform, called the Docker hub. Docker hub can be thought of as a huge repository of Docker images that are free to be downloaded and run. You can also create your own image and push it to the Docker hub.

03 Docker containers

Probably the most important component of the Docker ecosystem, containers can be thought of as running instances of Docker images. The image (which is going to be run as container) tells Docker what the container holds, what process to run when the container is launched and a variety of other configuration data. As mentioned in the previous step, images are read-only, but when the image is being executed (ie converted to a container), Docker creates a read-write layer on top of the image so that applications can run normally. Later, when the container is stopped, the top read-write layer is lost and the image is available as it is.

> “Containers isolate the runtime of the application you need to run, but all the containers use the same kernel”

Docker provides commands such as **run**, **start**, **stop**, **ps**, **rm** and so on, in order to interact with containers. You can type, **docker <command-name> --help** to get the help that's related to **<command-name>**.

Now that you have a clear idea of the important components and how they work together in Docker, let's install Docker and see the components in action. Note that I have used Ubuntu 14.04 as the base system for installation and demo purposes.

04 Installation

Docker installation is pretty straightforward. Firstly, log in to your system and check if you have wget installed, using the **which wget** command. If it isn't installed, run the commands **sudo apt-get update** and **sudo apt-get install wget**. After successfully installing wget, let's install Docker. Run the command **wget -qO- https://get.docker.com/ | sh**. The system prompts you for the root password, then downloads and installs Docker and its dependencies. To verify if Docker is successfully installed, you can run the command **sudo docker run hello-world**. This will download a test image and run it in a container on your system. If you take a close look at the output on the opposite page, you'll see the steps Docker followed to generate the message. This gives an idea of how components interact in Docker.

Docker Swarm

Docker swarm lets you create a native cluster for Docker – you can create a pool of Docker hosts. And since Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts. Though Docker Swarm is in the initial stages now, changes are expected to occur in how it will handle the backend.

Docker Compose

Docker Compose is a tool to help you create multi-container applications with Docker. This becomes especially important when you want to replicate the whole of your development environment – including CI servers, staging servers and so on. The process is fairly simple: just create a Dockerfile for your application, define the services required for your application in docker-compose.yml and then, lastly, run **docker-compose up**.

05 Create a Docker group

The Docker installation is successful and we have verified that using the **docker run** command. However, you may have noticed that we need to add **sudo** before running any Docker command, and therefore every time you run a Docker command in a new terminal, you need to enter the root password. This is because the Docker daemon binds to a Unix socket instead of a TCP port and the root user owns the Unix socket. To avoid using **sudo**, you can create a group called docker and add users to it; users here means the users supposed to run Docker commands. Now, when the Docker daemon starts, it sets the ownership of the Unix socket read/write to all users in the docker group.

To create the group and add a user, log in to Ubuntu as the user who needs to run Docker commands. Then run the command **sudo usermod –aG docker username** – don't forget to replace username with the actual username. Then log out of the system and log in again. Verify if Docker commands now work without **sudo** by typing **docker run hello-world**. If this works successfully, the group mechanism we added works. Now there is no need to enter the password every time you want to run a Docker command.

06 Base image

As discussed, Docker containers are created using images. An image can be basic, with nothing but the operating system fundamentals, or it can consist of a sophisticated pre-built application stack ready for launch. All this is possible because of Docker's layered approach to images. This provides an extremely powerful abstraction for building up application containers, where one step serves as the base for the next one, ie new tools, applications, content, patches and more, which form additional layers on the foundation (the base image).

A base image is the one without any further layers below; this is the most basic form of a Docker image that has just the operating system. The Docker hub has base images for all of the major Linux distros, but you are also free to create your own base image. To do so, you can use the **scratch** repository in the Docker hub, just use **FROM scratch** in the Dockerfile (more on Dockerfiles in the next step). Another way is using the **tar** command. Type

```
$ sudo debootstrap trusty trusty > /dev/null
$ sudo tar -C trusty -c . | sudo docker import - trusty
```

Here, the above command creates a tar file of the current directory and outputs it to STDOUT, where **docker import – trusty** takes it from STDIN and then creates a base image called trusty from it.

07 Dockerfile

As we saw in the previous step, creating a base image doesn't involve many Dockerfile instructions. However, as you add more and more layers to the base image, knowledge of Dockerfile instructions will become very important to you.

Let's start with its definition. A Dockerfile is a script composed of various commands (instructions) and arguments listed successively to automatically perform actions on a base image in order to create a new one. Creating a Dockerfile is as simple as listing the instructions one by one in a plain text file (named Dockerfile) and then calling the **docker build** command (in the same directory) to create the image. Typical Dockerfile syntax looks like this:

```
# Print "Hello World!"
RUN echo "Hello World!"
```

Dockerfile consists of two types of lines: comments starting with # and commands (with arguments) starting with an instruction (written in capital letters). Also, it is worth mentioning here that a Dockerfile should always start with the **FROM** instruction, meant to indicate the base image.

Building an image from a `Dockerfile`

Using the `docker commit` command is a pretty simple way of extending an image but it's a bit cumbersome and it's not easy to share a development process for images amongst a team. Instead we can use a new command, `docker build`, to build new images from scratch.

To do this we create a `Dockerfile` that contains a set of instructions that tell Docker how to build our image.

Let's create a directory and a `Dockerfile` first.

```
$ mkdir sinatra
$ cd sinatra
$ touch Dockerfile
```

If you are using Boot2Docker on Windows, you may access your host directory by `cd` to `/c/Users/your_user_name`.

Each instruction creates a new layer of the image. Let's look at a simple example now for building our own Sinatra image for our development team.

```
# This is a comment
```

08 Dockerfile instructions

There are about a dozen different commands that a Dockerfile can use to build a Docker image. Let's look at a few of the important Dockerfile instructions and their usage:

ADD – The ADD command needs two arguments: a source and a destination. It copies files from the source (on the host) into the container's own filesystem at the set destination. It can also take a URL as the source. Example:

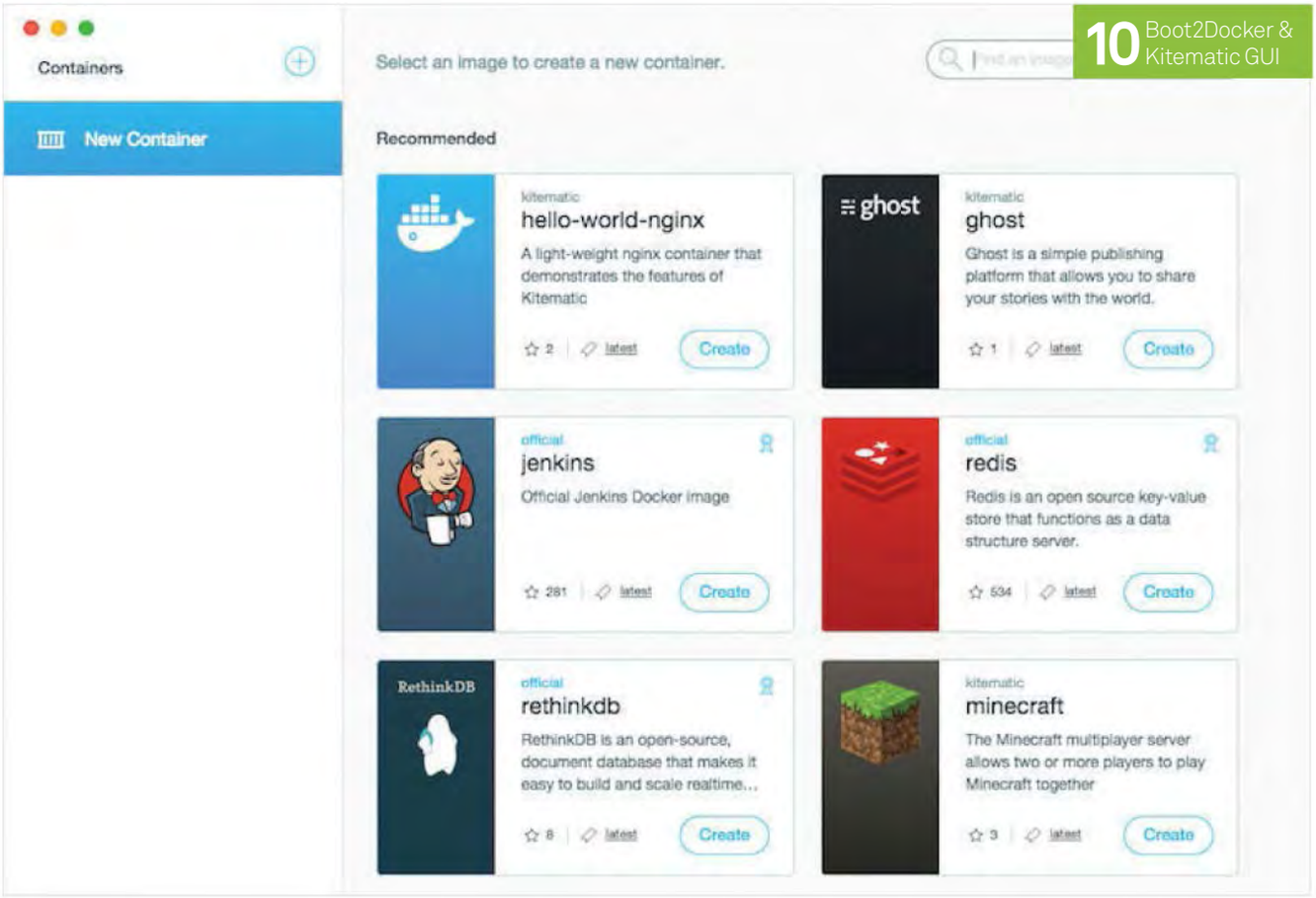
```
# ADD Usage
ADD /my_source_path /my_docker_dest_path
```

ENV – The ENV instruction is used to set the environment variables (one or more). These variables consist of key-value pairs that can be accessed within the container by scripts and applications alike.

```
# ENV Usage
ENV Default 6
```

FROM – This directive is the most crucial of those listed here. It defines the base image to use to start the build process. It can be any image, including the ones you have created before. It should be the first instruction in the Dockerfile.

```
# FROM Usage
FROM ubuntu
```



09 More Dockerfile instructions

RUN – This is the central executing directive for Dockerfiles. It takes a command as its argument and then runs it to form the image.

```
# RUN Usage
RUN apt-get update && apt-get install -y ruby ruby-dev
```

EXPOSE – This is used to associate a specified port to enable networking between the running process inside the container and the outside world (eg the host).

```
# EXPOSE Usage
EXPOSE 8080
```

MAINTAINER – This is a non-executing command, used to declare the author of the Dockerfile. Although not mandatory, it is good practice to include the author.

```
# MAINTAINER Usage
MAINTAINER Nitish <email@id.here>
```

VOLUME – This is used to enable access from a container to a directory on the host machine.

```
# VOLUME Usage
VOLUME ["/var/www", "/var/log/apache2"]
```

10 Boot2Docker and Kitematic GUI

As you may have noticed, Docker needs a Linux kernel to be able to run containers. This presents a blocking situation for non-Linux systems such as Windows or OS X. To overcome this, the Docker team created a CLI-based helper tool called Boot2Docker. The Boot2Docker installation package contains a VirtualBox virtual machine, the Boot2Docker tool and Docker itself. The virtual machine included in the package is a lightweight Linux VirtualBox image that provides the entire Linux-kernel related features required by the Docker daemon. So you can easily install Docker on your non-Linux PC.

However, the user interface still remains a concern for many users as they prefer the GUI instead of the CLI. Kitematic GUI is an open source project recently acquired by Docker. It automates the Docker installation and setup process, and provides an intuitive GUI for running Docker containers. When the Kitematic GUI launches, you will be presented with curated images on the home screen that you can run instantly. You can search for any public images on Docker Hub from Kitematic just by typing in the search bar. Note that Kitematic is currently available for Mac only, with Windows support expected to launch soon. ■

Above Boot2Docker circumvents blocking on non-Linux systems

Deploy software to Docker containers using Deis

Deis is an application platform that helps you deploy and scale applications as Docker containers

Resources

Docker
docker.com

Deis
deis.io

It is a thing of the past to develop code and wait for the operations team to deploy it with your fingers crossed. DevOps is slowly gaining steam as the preferred mode of software development and deployment. DevOps methodology and its automation tools make it possible to deploy and test code in almost any kind of environment. Deis is one tool that lets you deploy a project written in any language or framework with a simple **git push**. You can also use Deis to promote existing Docker images through a continuous integration pipeline.

Being a deployment tool, Deis integrates very well with Docker. All the applications you can deploy via Deis are distributed as Docker containers. Deis itself runs as a combination of several Docker containers. Also, Deis is designed to run applications adhering to the Twelve-Factor app methodology. A Twelve-Factor app is a methodology to build apps that scale up easily; lessons learned while using the Heroku platform to build SaaS apps are the inspiration behind it.

Deis runs on CoreOS, a new minimal distro designed to run containerised infrastructure stacks. So before you can install the Deis platform and client, you'll need a cluster with CoreOS installed. No need to worry though, as CoreOS can be installed on almost any infrastructure.

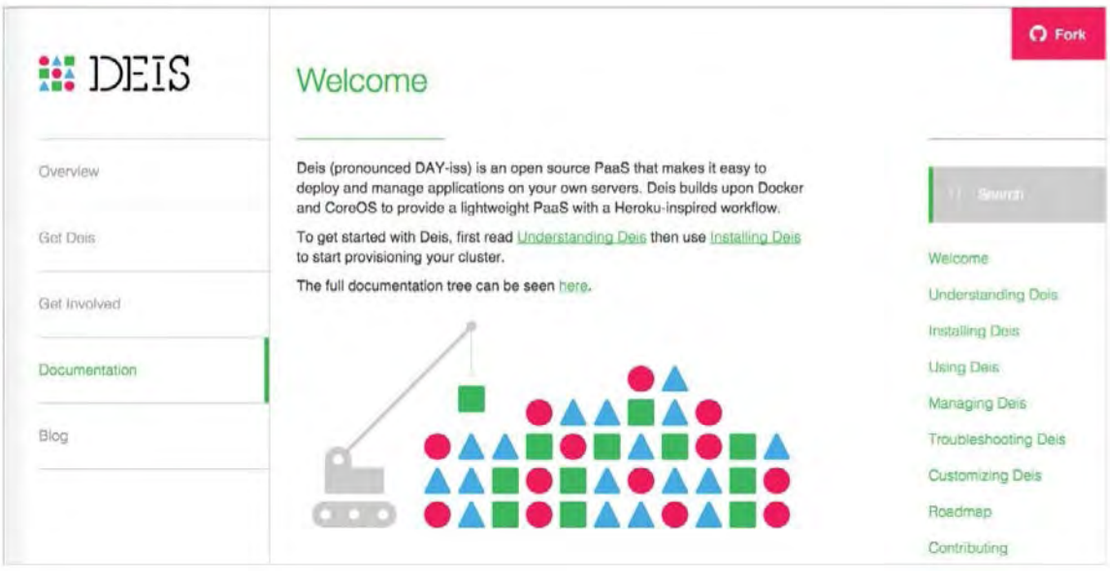
01 Deis architecture

Deis is based on a service-oriented architecture, with all of its components clearly segregated from one another, each having a clearly defined role in the larger scheme of things.

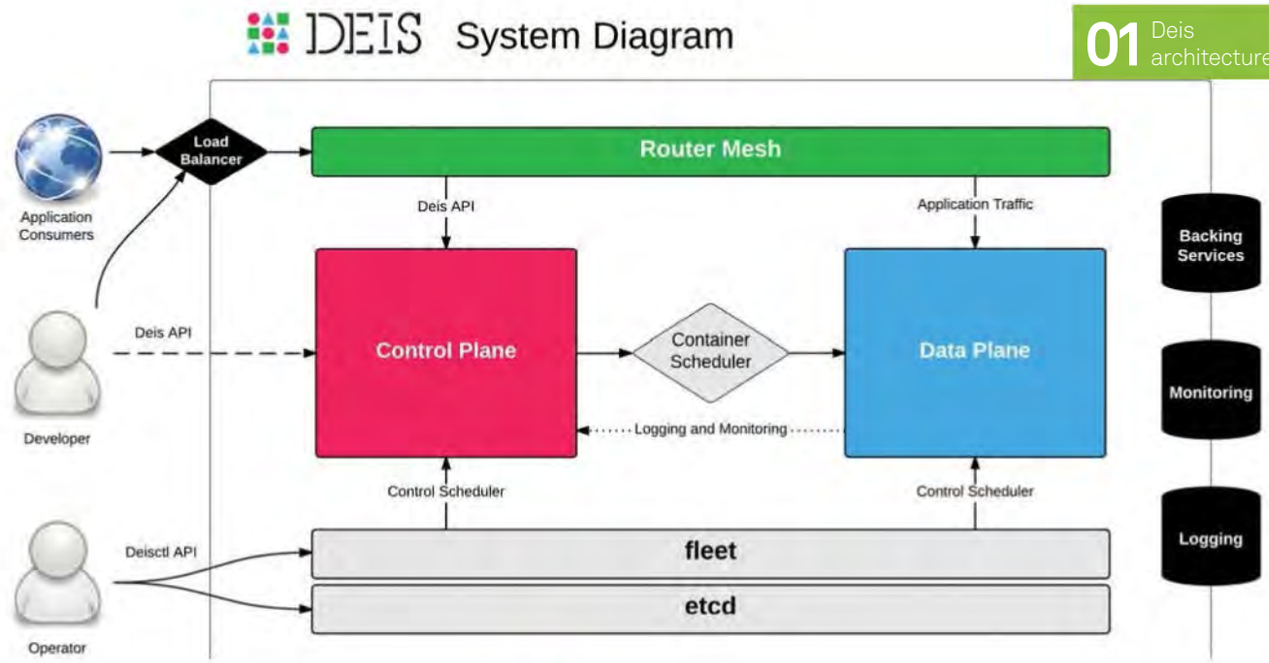
The various components are categorised into three major groups: data plane, control plane and router mesh. We will take a look at them in detail in the following steps, but first let us understand how these components communicate and get things done.

As the name suggests, the control plane handles the management functions, such as storage, managing configurations and so on. The data plane runs the actual application code in the form of Docker containers and the two communicate with each other via a scheduler.

The router mesh is used to handle the incoming web traffic. Since it is usually connected to the public Internet, it is often coupled with a front-end load balancer. The incoming traffic is of two types: developers accessing the control plane using the Deis API requests that are routed to the control plane, and the traffic meant for the application software that's routed to the data plane.



Right Deis can take code from your Git repo and publish it as Docker containers on your server cluster automatically



01 Deis architecture

02 Control plane

The control plane is a set of components that handle the overall Deis platform. As mentioned in the previous step, developers can interact directly with the control plane using the Deis API.

The major components of the control plane are: the controller, builder, store, registry, database and logger. While these names seem self-explanatory, they can cause a wrong perception so let's take a look at each one.

The controller component is a HTTP API server. It receives the external Deis API request and triggers corresponding actions. The builder component receives the incoming **git push** requests. After authenticating the user credentials, it builds a new Docker image from the updated Git repo and pushes it to the registry component. Finally, it triggers a new release via the controller component. The storage component is based on Ceph clusters and provides simple APIs to store all the stateful components. The registry component is actually a Docker registry used to hold the images and other configuration data. The database component is a PostgreSQL database used to hold stateful data. The logger component is a logging server used to hold all the aggregated logs from the data plane.

03 Data plane

We have already seen the controller and builder components of the control plane that interface with the developers and Git repos respectively. As and when there is new code pushed to the Git repo, the Deis control plane swings into action, resulting in spawning containers with the latest code. These containers are run in the data plane.

Here is how it works: once the controller component (from the control plane) identifies new source code, it triggers the scheduler to start containers. The scheduler decides which container should be running at which machine and starts the containers. These containers are part of the data plane. Other components of the data plane are the publisher and logspout. The publisher is responsible for publishing the containers so that they are accessible to the outside world via the router mesh. The logspout, on the other hand, collects logs from running containers and then sends them to the logger component (control plane).

> Deis is one tool that lets you deploy a project written in any language or framework with a simple **git push**

04 Achieve scale by isolating the planes

We have already learned about the three major parts of the Deis platform: data plane, control plane and the router mesh. Now we need to understand the topology to be used for small and large deployments.

For small deployments, you can run the entire platform on just three servers, ie just use the default settings. But as the deployment grows larger, the best way to scale your Deis platform is to isolate the control plane and the router mesh from the data plane. You then limit the control plane and router mesh to a small, fixed number of nodes and let the data plane be scaled as per the requirement. This is because the control plane and the router mesh have no significant need to scale (they just pass on requests to other layers), while the data plane actually runs the application containers – it is the most resource-hungry part of Deis.

Deis clusters

Deis is a highly available distributed system, which means that Deis components and your deployed applications will move around the cluster onto healthy hosts as hosts leave the cluster for various reasons. So you should have ample spare resources on any machine in your cluster to withstand the additional load of running services for failed machines. It is recommended that each node should have at least 4 GB RAM and 40 GB hard disk space.

Try Deis

An automated deployment of a Deis cluster on AWS is provided at **try.deis.com**. Just create a user using the AWS IAM application and assign the 'AdministratorAccess' policy. You can then use the credentials at **try.deis.com** to provision CoreOS cluster and deploy your applications on the cluster using Dockerfiles.

05 Deis installation preconditions

Deis needs a cluster of at least three CoreOS-based machines, but before you can create the cluster, let's start with the preconditions. First of all, you will need the Deis source code on your system, as only then can the further commands be run on your system. You can download the source code from GitHub using:

```
$ git clone https://github.com/deis/deis.git
$ cd deis
$ git checkout v1.10.0
```

It is recommended to use the latest release instead of the default. The next step is to generate an SSH key because Deis communicates with remote machines using a SSH tunnel. Create a keypair with:

```
$ ssh-keygen -q -t rsa -f ~/.ssh/deis -N '' -C deis
```

Now create a discovery URL. A discovery URL helps in linking the various instances together. It is mandatory to create it before provisioning a CoreOS cluster. You can do this using:

```
$ make discovery-url
```

Finally, you can now create a CoreOS cluster. Note that you'll first need to decide the cluster provider, eg Amazon Web Services (AWS), DigitalOcean, Linode, OpenStack, Google's Compute Engine and so on. You can also use bare metal servers. Next, we'll learn how to provision a cluster on AWS.

06 Deis control utility installation

Now that the preconditions are met, let us install the Deis control utility. The Deis control utility, or in other words deisctl, serves as the single point of interaction with the Deis control and data planes.

To install deisctl, first navigate to the folder you would like the binary to be installed. Then execute the command:

```
$ sudo curl -sSL http://deis.io/deisctl/install.sh |
  sudo sh -s 1.10.0
```

As you may have noticed, this installs deisctl version 1.10.0 to the current directory and downloads the matching Deis system unit files to schedule the components. Now we need to link desictl into usr/local/bin, so that it gets added to the $PATH:

```
$ sudo ln -fs $PWD/deisctl /usr/local/bin/deisctl
```

It is important to note here that we installed Deis version 1.10.0 and then used the same version of deisctl. It is mandatory to do so. Once you install deisctl, cross-check the version using the command:

```
$ deisctl --version
```

07 Cluster provisioning

First, install the AWS command line interface awscli to be able to work with the Amazon API, and PyYAML for the Deis AWS provision script to run. Type the following:

```
$ pip install awscli
$ pip install pyyaml
```

Now set your AWS credentials using the AWS command line configuration tool. Just type:

```
$ aws configure
```

Note that these credentials are the not the id/password you use to log in to the AWS web console. You need to create the AWS access key ID and AWS secret access key using the AWS IAM application. After creating a new user in IAM, don't forget to attach the 'AdministratorAccess' policy to it. The next step is to upload the keys we created previously. Type:

```
$ aws ec2 import-key-pair --key-name deis --public-
  key-material file://~/.ssh/deis.pub
```

By default, this setup provisions three servers. If you'd like to add more, you can do it by setting:

```
$ export DEIS_NUM_INSTANCES=5
```

Now you can personalise the cloudformation.json file. Navigate to the directory /contrib/aws in the Deis repository and look for the file cloudformation.json. You can add any of the parameters defined in the file deis.template.json to cloudformation.json and edit it as per your needs.

Finally, you can execute the provisioning script to spawn the CoreOS cluster. The script is in the same directory we navigated to. Just run:

```
$ ./ provision-aws-cluster.sh
```

08 Deis platform installation

This is the final step in the Deis installation process. Here we will install the Deis platform on your local system. Get started by checking if the SSH agent is running and then selecting the keys that we created previously:

```
$ eval 'ssh-agent -s'
$ ssh-add ~/.ssh/deis
```

The next step is to find the public IP of one of the nodes that we provisioned in the previous step. Then export it to the DEISCTL_TUNNEL environment variable:

```
$ export DEISCTL_TUNNEL=<IP_Address>
```

Replace **IP_Address** with the public IP address of the node. To check if deisctl can communicate with the AWS nodes, type:

```
$ deisctl list
```

Add the SSH key to Deis in order to connect to the cluster:

```
$ deisctl config platform set sshPrivateKey=~/.ssh/deis
```

05 Deis installation preconditions

Left Check that you're getting the latest version by referring to the main GitHub page first

You also need to set the domain name under which you are planning to deploy applications:

```
$ deisctl config platform set domain=example.com
```

Finally, install the platform itself:

```
$ deisctl install platform
```

Note that you must follow the steps mentioned here in the exact same order, as other ways may end up causing you problems with the **install** command.

09 Deis usage

Now that the platform is installed, let's see how to use Deis. First, you will need to install the Deis client and then register a user. To install the client, type the following in the terminal window:

```
$ curl -sSL http://deis.io/deis-cli/install.sh | sh
```

The installer puts Deis in your current directory, but you should move it somewhere in your $PATH. Use the below command:

```
$ ln -fs $PWD/deis /usr/local/bin/deis
```

To register a new user, you need to use the **deis register** command. Also, you need to use the domain name that we set previously and then append it with **deis.<domain_name>**. For example, if the domain name we set is **example.com**, the URL to be used with **deis register command** will be **deis.example.com**.

```
$ deis register http://deis.example.com
```

Note that the first user to register automatically will receive the superuser privileges. Additional users who register become ordinary users. After logging in, the new user should upload their SSH public keys, in order to be able to use **git push** to deploy applications to Deis. You can do this using :

```
$ deis keys:add
```

Users can log in using:

```
$ deis login http://deis.example.com
```

...and log out using:

```
$ deis logout
```

10 Deploy applications using Dockerfiles

To be able to deploy applications to your CoreOS cluster via Dockerfiles, you will need to first clone the application from your Git repo.

```
$ git clone https://github.com/deis/helloworld.git
$ cd helloworld
```

Note that the Dockerfile will have to conform to a few requirements as listed below:

- Dockerfile should **EXPOSE** only one port
- The port must listen for HTTP connections
- A default CMD must be set for running the container

You can create an application using the **deis create** command on the controller. Finally, use **git push deis master** to deploy your application. Deis checks for any changes in the source and then builds a Docker image using the Dockerfile. After building the image, it is deployed in the Deis data plane (running in the CoreOS cluster). ■

Run science experiments on the ExpEYES kit

ExpEYES is a cheap digital oscilloscope with a signal generator and other features, making it the ultimate tool for electronics

Resources

Raspberry Pi
Model B

ExpEYES kit
bit.ly/1AR15dz

ExpEYES is a relatively unheard of but very impressive hardware and software platform for science and electronics experimentation, as well as a useful electronic probing tool for makers and professionals alike. It is also open source on both the hardware and software sides, which makes it affordable and versatile.

ExpEYES is billed as a science and experimentation kit but really it is much more than that – it is a fully-functioning four-channel digital oscilloscope with an impressive array of features. ExpEYES ships with a wealth of online documentation in a variety of formats (graphics, user guides, web content), including upwards of 50 suggested experiments, and the kit itself contains all of the hardware required to play with the interesting science of electronics contained within the guide material. The aim is to enable the learning of what can be complex concepts of electronics in an easy and affordable way, without getting bogged down in the arcane details. Paired with our favourite little single-board computer, the Raspberry Pi, you have an extremely powerful and affordable device in your hands.

01 Get the parts

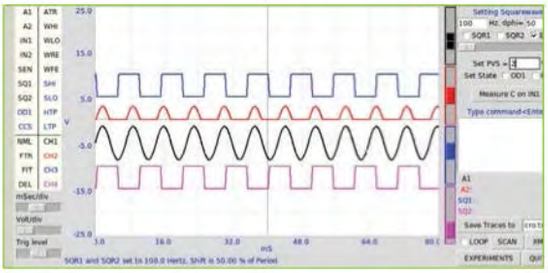
ExpEYES is available to purchase from a variety of online vendors, including CPC (**http://cpc.farnell.com**), for around £50. It is possible to get the kits slightly cheaper from India or China (see **bit.ly/1H38EFC** for other vendors worldwide), however it's likely to end up costing more due to higher shipping rates as well as potential import fees and duties.

Left The kit itself is highly portable and great for taking down to Jams and hackspaces

02 Open it up

The ExpEYES kit contains everything you need to get underway, with over 50 documented experiments from the ExpEYES website. The only other item that may come in handy is a breadboard. You will also need a Raspberry Pi or other computer with a USB port in order to run the digital oscilloscope software and connect to ExpEYES.

> "It pays dividends to make sure that your operating system is updated to the latest stable version, as this can save you a lot of hassle"



03 What's inside?

As you may have guessed, the ExpEYES kit includes the main ExpEYES USB digital oscilloscope, but it also contains a wide range of other hardware including a DC motor, magnets, LEDs, coils, piezoelectric discs, wiring, a small screwdriver for opening the screw terminals and more. You also get a live CD which contains all the ExpEYES software and documentation ready to go on a bootable disc.

04 What can it do?

The chip at the heart of ExpEYES is an AVR ATmega16 MCU (microcontroller unit), running at 8 MHz coupled to a USB interface IC (FT232RL). These are low-cost but provide good value for money. As we have already mentioned, ExpEYES is therefore capable of acting as a four-channel oscilloscope but also has a built-in signal generator, 12-bit analogue resolution, microsecond timing resolution and a 250 kHz sampling frequency. At this price point, that's an impressive set of features and certainly accurate enough for anything that is not mission critical (like learning, hobby projects, quick readings and so on).

05 Using the live CD

Perhaps the easiest way to get up and running with ExpEYES (if you have a computer with a CD drive) is to use the live CD which is included in the ExpEYES kit. Making sure that you are booting into the live CD from your BIOS boot menu, you should then be greeted with a Linux-based desktop. Plug in your ExpEYES by USB and you can open the software from the menu by going to Applications>Science>ExpEYES-Junior. Alternatively, you can run it from a terminal window using:

```
sudo python /usr/share/expeyes/eyes-junior/
croplus.py
```

06 Update your Raspberry Pi

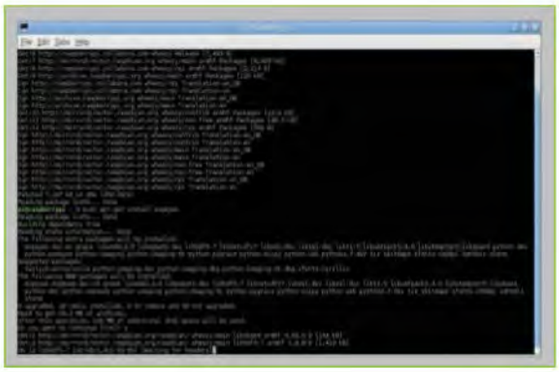
As with almost every project you undertake on the Raspberry Pi, it pays dividends to make sure that your operating system is updated to the latest stable version, as this can save you a lot of hassle further down the line. To do this, open an LXTerminal session and then type **sudo apt-get update**, followed by **sudo apt-get upgrade –y**, and then wait patiently for the upgrade process to complete.

Other supported platforms

The ExpEYES software is mainly written in Python. This means that the core software to run your ExpEYES device is quite platform-agnostic – if the device can run a Python interpreter and has a Python module enabling it to access the serial port then it will work with ExpEYES. If you visit the ExpEYES website, there is a page that explains how to install the software on Linux and Windows – **www.expeyes.in/software-installation**. In addition, a native Android app will enable your ExpEYES to work with any Android device that has USB on the go (OTG) capability.

ExpEYES & PHOENIX

ExpEYES was developed by Ajith Kumar and his team as part of the PHOENIX (Physics with Homemade Equipment and Innovative Experiments) project, which was started in 2005 as a part of the outreach program of the Inter-University Accelerator Centre (IUAC) in New Delhi, India. Its objectives are developing affordable laboratory equipment and training teachers to use it in their lesson plans.



07 Install the software

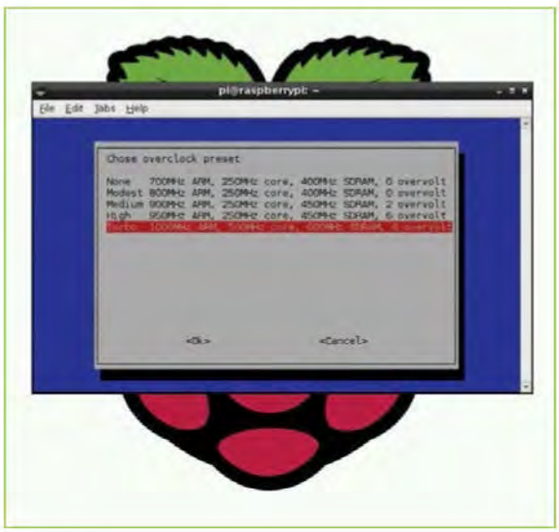
Due to efforts of community member Georges Khaznadar, there are DEB packages available for the ExpEYES software that should work perfectly on Debian, Ubuntu, Linux Mint and, of course, Raspbian. These are also included in the official Raspbian repositories, so all you need to do to install the ExpEYES software is to open an LXTerminal session on the Raspberry Pi and then run the following commands:

```
sudo apt-get update
sudo apt-get install expeyes
```

08 Install dependencies

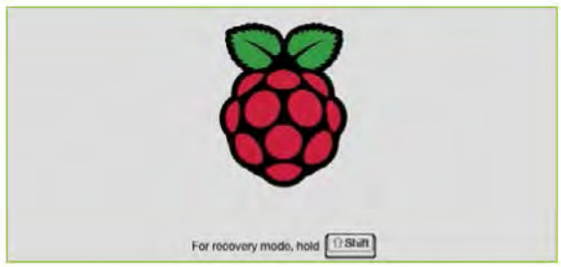
ExpEYES has a number of dependencies that are required for it to run under Linux, as well as a number of other recommended libraries. During the installation undertaken in Step 7, the dependencies should be installed by default. However, to avoid any problems later, you can run the following command in order to make sure that they are all installed:

```
sudo apt-get install python python-expeyes python-
imaging-tk python-tk grace tix python-numpy python-
scipy python-pygrace
```



09 Overclock your Raspberry Pi (optional)

The ExpEYES software will run fine on a Raspberry Pi with default settings, however it can be slow to respond if you are using a Model A, B or B+. We recommend using a Model 2B, but if you don't have one, overclocking your Pi would be advisable (you can overclock your 2B as well if you want it to run a bit faster). Open an LXTerminal session and type **sudo raspi-config**. In the menu, select the option '7 Overclock'. Click OK on the following screen and then select Turbo. Click OK and you should see some code run. Once this completes, press OK again and then you are brought back to the main raspi-config window. Select Finish in the bottom right and Yes to reboot your Raspberry Pi.



10 Overclocking continued

Overclock can sometimes cause instability on your Raspberry Pi or an inability to boot at all. If this happens you can press and hold the Shift key on your keyboard once you reach the above splash screen to boot into recovery mode. You can then redo Step 7 at a lower overclock setting and repeat until you find the highest stable setting.

11 Resistance of the human body

An interesting experiment for your first time using an oscilloscope it to measure the resistance of the human body over time. This is easy to accomplish with just three bits of wire and a resistor (200 kOhm). On the ExpEYES, connect a wire between A1 and PVS, connect the resistor between A2 and ground, and connect an open-ended wire out of both PVS and A2. Plug in your ExpEYES and open the control panel, then drag A1 to CH1 and A2 to CH2, and set PVS to 4 volts. You can then pick up one of the open-ended wires in each hand and watch the response on the ExpEYES control panel.

12 Run the maths

From the output plot, you should find that the input on CH1 is coming out at 3.999 volts (which is great because we set it to be 4!). The voltage on A2 (CH2) is showing as 0.9 volts for us, which implies that the voltage across the unknown resistor value (your body) is 4 – 0.9 = 3.1 volts. Using Ohm's law (V=IR), we can then calculate the current (I) across the known resistor value: voltage ÷ resistance = 0.9 ÷ 200,000 = 0.0000045 amps = 4.5 uA (micro amps). Using this value we can then calculate the resistance of the body using the same Ohm's law equation in reverse: voltage ÷ current = 3.1 ÷ 0.0000045 = 688889 ohms = 689 kΩ. This is a surpisingly high value, however the resistance of the human body depends hugely on how dry your skin is and a large number of other factors (body resistance is usually in the range of 1,000 to 100,000 ohms).

Above There's a great range of experiements for you to try inside the ExpEYES documentation over at: **bit.ly/1E7hdYy**

13 Use the Python library

The ExpEYES team have built a custom Python library for the device. This is slightly harder to use than the GUI and not as pretty, but it enables a lot more versatility as well as the capability to use ExpEYES functionality within your Python scripts. If you have followed the installation instructions above, all you need to do is import the Python module and then initialise a connection to the ExpEYES using:

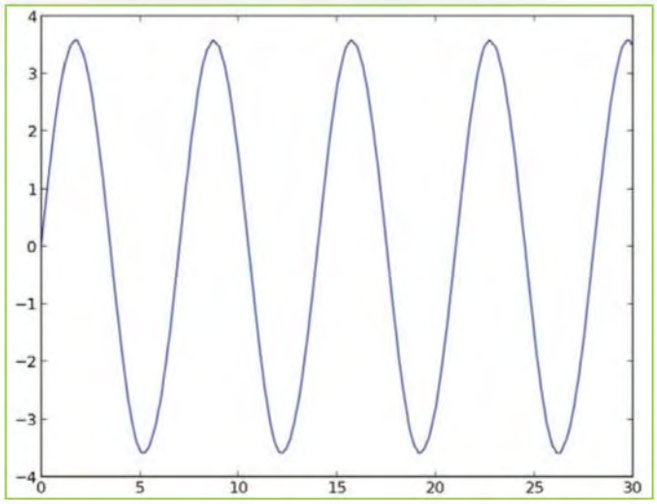
```
import expeyes.eyesj
p=expeyes.eyesj.open()
```

> "A digital storage oscilloscope is a useful tool in any engineer or hacker's toolbox, as it enables you to get insights into your projects that aren't possible with visual checks"

14 The Python library (continued)

Now we will plot a sine wave using the ExpEYES and PyLab libraries. On the device, connect OD1 to IN1 and SINE to A1 with some wire. Run the following code and you should see that a sign wave has been plotted.

```
import expeyes.eyesj
from pylab import *

p=expeyes.eyesj.open()
p.set_state(10,1)
print p.set_voltage(2.5)
ion()      # set pylab interactive mode
t,v = p.capture (1,300,100)
(plot t,v)
```



15 Further experiments

This tutorial has shown you just a single example of the documented ExpEYES experiments available at **http://expeyes.in**. There is a wide variety of different techniques and phenomena explored in those experiments, so it is highly recommended to get your hands on an ExpEYES kit and work through them. Running through those examples as a beginner will give you a much deeper understanding of electronics.

16 The verdict

A digital storage oscilloscope (plus extras) is a useful tool in any engineer or hacker's toolbox, as it enables you to get insights into your projects that aren't possible with just visual checks or using a multimeter. Whilst no £50 oscilloscope will compare to expensive professional units, this is a great entry-level product as well as a versatile, portable USB device with multiplatform support for when you just can't be lugging around a 10 kg, £1000+ scope. ■

Program a Parrot AR.Drone's flight path

Use Python to program a fixed flight path for your Parrot drone and then modify it for better control

Resources

python-ardrone
bit.ly/1eQ7XTz

pygame
pygame.org/news.html

Firmware 1.5.1 for the Parrot AR.Drone

If you've ever tried to properly fly a Parrot AR.Drone with an iOS device, you'll probably know how inaccurate it can be. With the touch controls, it's not always obvious if they're working or not, and although with a lot of practice you can get used to it, it's never quite exact.

The drones are quite smart, though. There are several functions you can use to make one take off, land and hover, using various sensors to enable you to do this safely. All of these, along with the flight controls, are just code when you get down to it.

While it's not quite an open API, people have managed to hack together libraries that access the AR.Drone, enabling you to send all the same commands as the official apps. There are many ways you can then use this, but we're interested in creating a programmable route for the drone.

We'll be using a library for Python that enables us to do just this, and even lets us see video from the Parrot as well if it is set up correctly.

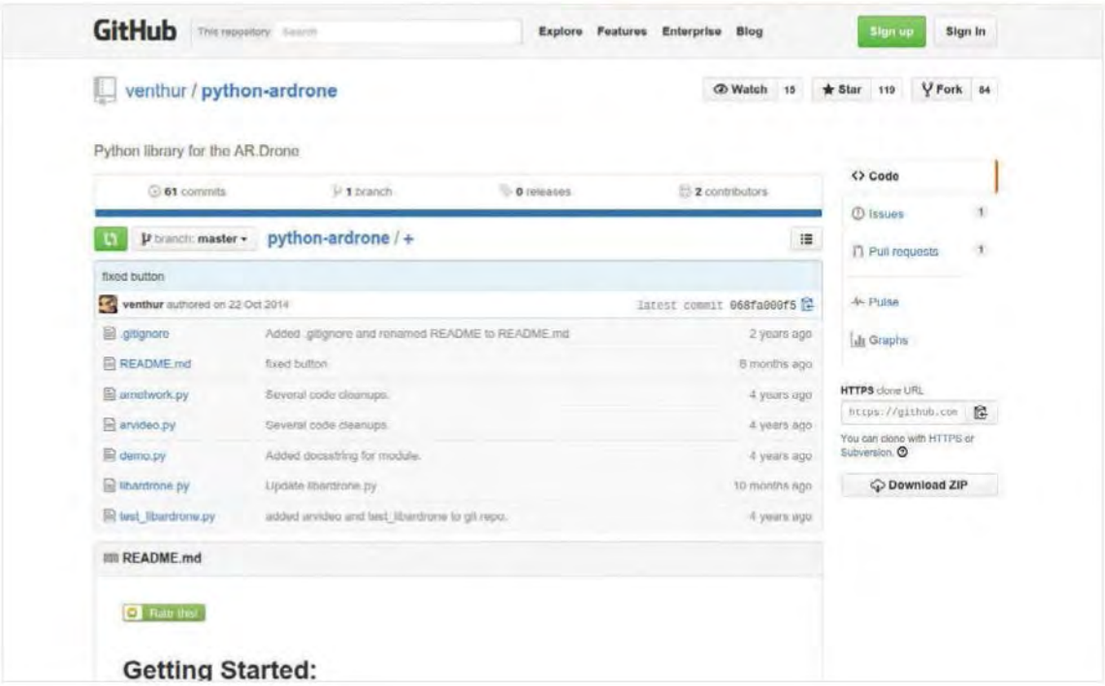
For this tutorial, we're going to be using the python-ardrone module created specifically to program drones via Python. It's not available from repos or pip, so you will need to grab it directly from GitHub. You can find it here:

https://github.com/venthur/python-ardrone

Click the 'Download as Zip' button to do just that and save it to a working folder for this project. Unzip it directly into this folder so that we can use the various files as modules, which will then connect to the other files in order to connect to the drone and so on.

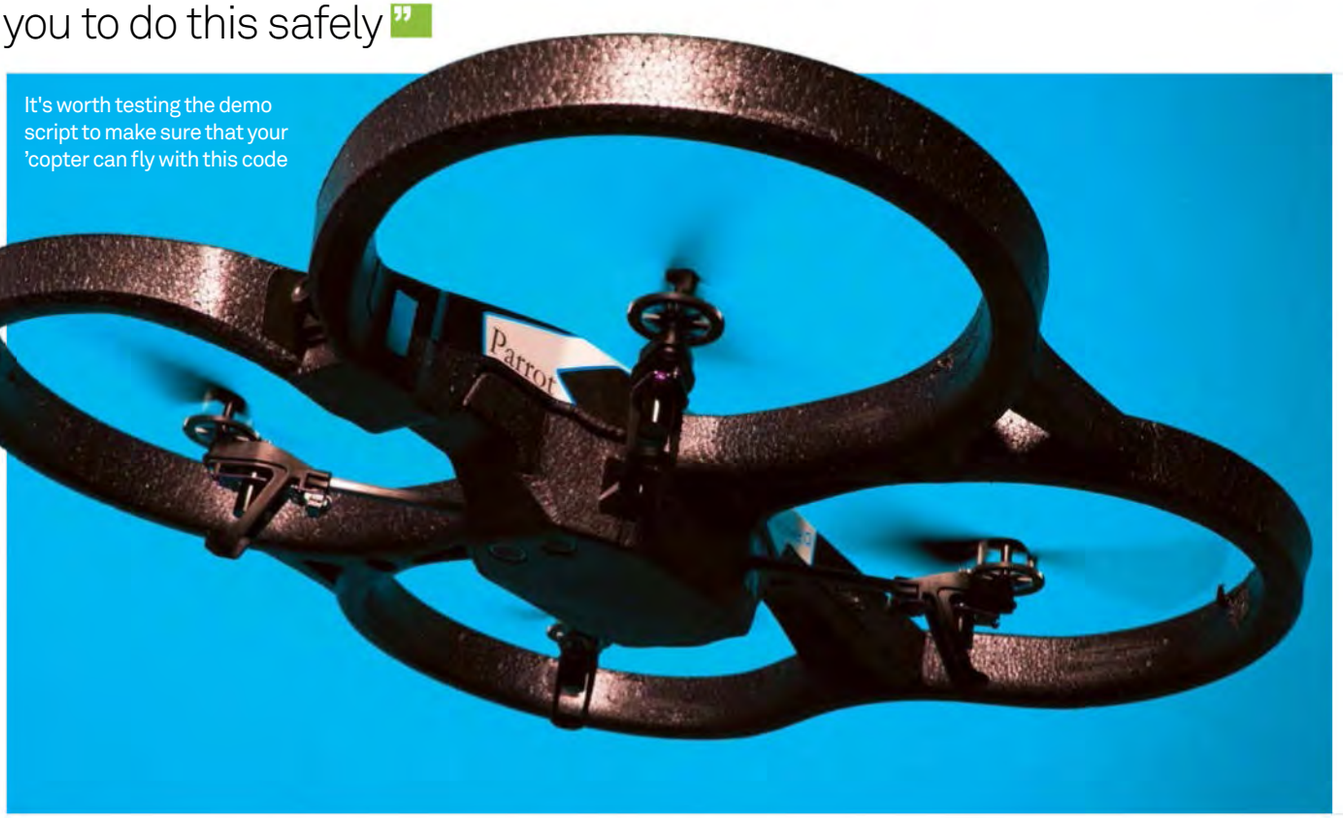
You'll also need to install pygame (**pygame.org/download.shtml**) for some of the tests we're going to run, so either get it directly from the website or install it to your system from the repositories by opening the terminal and using:

```
$ sudo apt-get install python-pygame
```



Right Download python-ardrone off the GitHub website

> There are several functions you can use to make it take off, land and hover, using various sensors to enable you to do this safely



It's worth testing the demo script to make sure that your 'copter can fly with this code

Pygame enables us to create a window from which we can view the camera output, although it's a little tricky to add in a very simple route program – we'll cover why that is later on.

Once everything is installed and ready, open up IDLE and create a new Python file, naming it 'route.py'. Put it in the working folder so that it can interact with the different ardrone modules we downloaded earlier.

Before we start coding and testing, it's worth actually making sure your quadcopter will fly using this code. You'll first want to connect to it via Wi-Fi on your PC, so with the AR.Drone itself, hit the Wi-Fi sync button and connect to it with your computer's Wi-Fi manager. You would normally do this to upgrade firmware and software, but the connection still works the same as it does on a phone and signals to control it can be sent back and forth.

Once connected, go back to IDLE and click on File>Open and navigate to our project folder. From there select 'demo.py', which is a test script that lets you very simply control your drone using keyboard keys. It also displays the video feed as well, so you can see how pygame will handle that if you decided to use it. With the demo running, you can use the following keys to control the drone – it's worth a test to make sure everything is working properly:

RETURN	Takeoff
SPACE	Land
BACKSPACE	Reset (from emergency)
A/D	Left/right
W/S	Forward/back
1 - 0	Speed
UP/DOWN	Altitude
LEFT/RIGHT	Turn left/right

If you're having problems, it's worth checking the firmware version of your drone. To update the firmware, you need to be connected to it via Wi-Fi, and then connect to it via an FTP client. In FileZilla, you can do a quick connect to the IP 192.168.1.1 using port 5551. You might need to search around for a firmware 1.5.1 file, but once you get it you just need to upload the file to the drone. Now disconnect and reboot the drone for the firmware to take affect.

Once you can satisfactorily circle the living room, land your drone and then exit the code by hitting the Escape key. You can close the demo code for now, although it may be useful to refer back to in the future for modifying the path we create.

Other libraries

This is just one library that works with the AR.drone in Python. There are a few others created by the community, and even one you can get from Parrot that works with it. They all have their different ups and downs, but this one is quite easy to start with and works well with pygame.

The demo code creates a while loop that continually checks to see if specific keys are pressed, which is also part of the pygame module, and then activates any function that the key press is tied to. W was linked to forwards and therefore used **move_forward()** in the module. It requires you to hold it down to actually move any useful distance, though, and when nothing is being pressed it defaults to hover mode.

Let's build a basic script that will show how to use the module. In the blank 'route.py' document, enter the following:

```
import libardrone
```

```
drone = libardrone.ARDrone()
drone.takeoff()
drone.hover()
drone.land()
drone.halt()
```

As the ardrone module is quite readable, it's quite simple to figure out what this would do. We've imported the ardrone library, then used the takeoff function so that it actually takes off, told it to hover and then land again. The halt function disconnects everything, making the exit clean. Each part of this code will execute pretty quickly, so if you want it to hover for any particular length of time then you'll have to start adding sleeps and loops, but we'll get to that. For our little starter route, we're going to make something very simple: it will take off, rise in height a little and then fly a complete square around an area.

We'll have it rotate left and right on the top-left corner, and do a full 360 on the top-right. Refer to the diagram below for a better visualisation of what we are talking about.

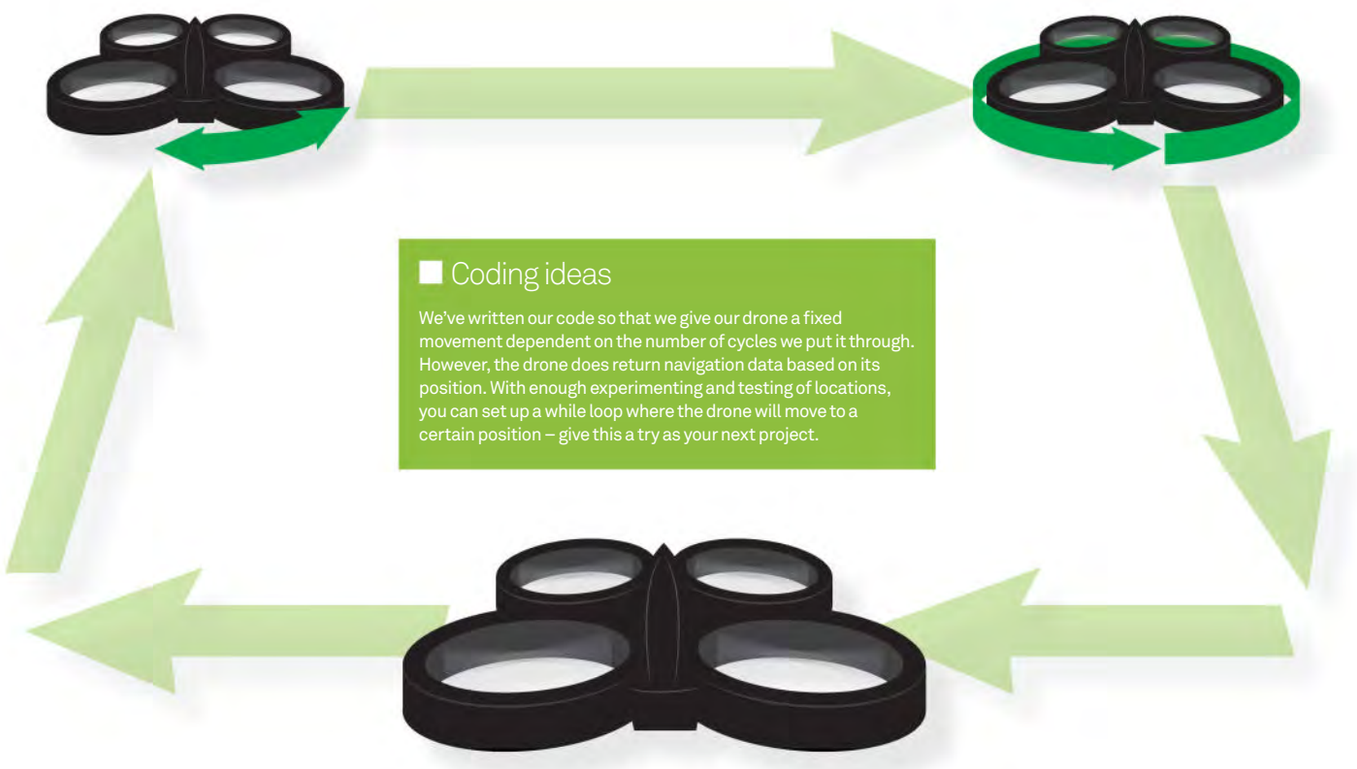
Once you've understood what we plan to do, it's time to plan out the code. One of the big things is making sure the code tells the drone to actually perform any action a number of times that makes it useful in real life. The exact number will require testing on your part, but we'll handle the code bit with a for loop:

```
for i in range(x)
```

... where x is the number of times you want it to repeat a function (or you can instead put in a variable with a preset or changing function). If you refer to our full code listing, you can see how it's used in our specific script. For example, for the first move to the left we've done:

```
for i in range(5):
    drone.move_left()
```

Below The simple starter route we're programming here



Coding ideas

We've written our code so that we give our drone a fixed movement dependent on the number of cycles we put it through. However, the drone does return navigation data based on its position. With enough experimenting and testing of locations, you can set up a while loop where the drone will move to a certain position – give this a try as your next project.

> When creating this code, make sure you write and test each step as often as you can - this way you can get the numbers correct

This will execute **move_left()** five times. Once that's complete it will go onto the next command, working through each numerous iteration until the code ends. If you want the drone to hover for a while before doing the next move, you'll have to do the same thing and keep calling **drone.hover** for a period of time.

The rest of the new code chunks are simple: **def main()** is the function of the script, and when this code is run on its own, it will execute the function main using the final part of code: **if __name__ == '__main__'**. This means you can use it as a module in other code if you want to create an interactive experience with your drone in the future.

When creating this code, make sure you write and test each step as often as you can – this way you can get the numbers correct. We've used five iterations in most of the code, but it won't move very far at all under that.

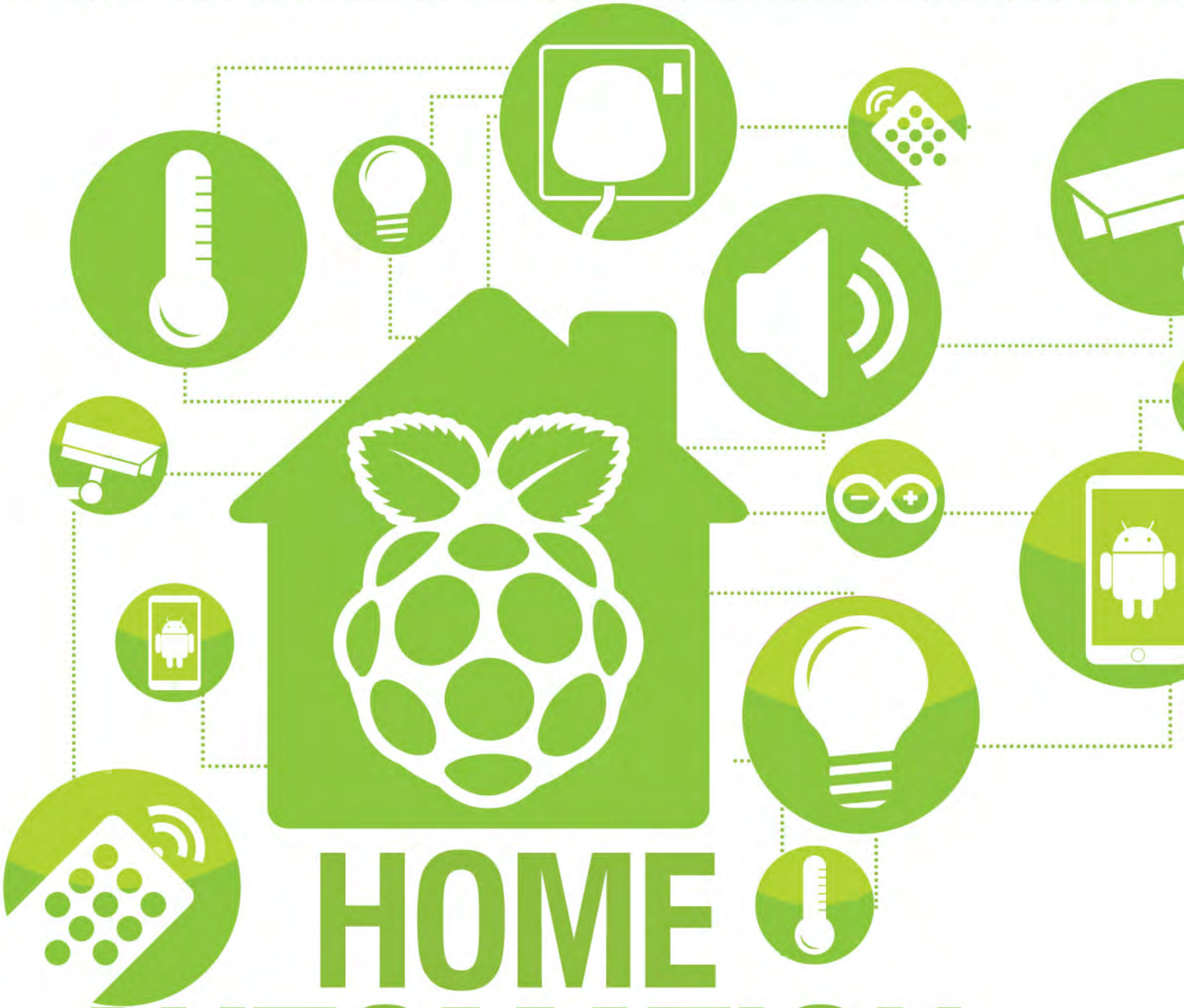
For expanding the code in the future, here are all the control commands you can use:

takeoff()	Launch the drone
land()	Land the drone
hover()	Keep the drone hovering (you could use a while loop to keep this active)
move_left()	Move the drone to its left
move_right()	Move the drone to its right
move_up()	Ascend the drone
move_down()	Descend the drone
move_forward()	Move the drone forward
move_backward()	Move the drone backwards
turn_left()	Make the drone rotate left/counter-clockwise
turn_right()	Make the drone rotate right/clockwise
reset()	Emergency stop
trim()	Level off the drone
set_speed()	Set the speed percentage, with 1 being 100%
halt()	Stop communications, end all relevant processing threads
move()	Advanced movement, refer to the libardrone code on how to use

The above commands are more than enough to get you up and running , and you'll discover that there is plenty you can do with drones. We would recommend checking out legal requirements for where and how to fly, though.

Full Code Listing

```
import libardrone

def main():
        drone = libardrone.ARDrone()

        # set initial speed
        drone.speed = 0.5

        # take off
        drone.takeoff()

        # go higher
        drone.move_up()

        # go to the left
        for i in range(5):
                drone.move_left()

        # go forward
        for i in range(5):
                drone.move_forward()

        # turn the drone to move the camera
        for i in range (3):
                drone.turn_right()

        # turn it back again
        for i in range (3):
                drone.turn_left()

        # go right to complete a square
        for i in range (10):
                drone.move_right()

        # full 360
        for i in range (10):
                drone.turn.left()

        # return to start
        for i in range (5):
                drone.move_left()
                drone.move_backward()

        # land the drone
        drone.land()

        print "I have landed. I hope you enjoyed the
flight. I will now shut down"

        drone.halt()

if __name__ == '__main__':
        main()
```



HOME AUTOMATION WITH YOUR PI

Your smart home setup

Remote control sockets

Energy saving and green houses are a big thing right now, and you can buy power strips that will shut down every socket based on the draw from a single socket. This isn't always accurate, though, and being able to manually control the socket is not always easy if it's hidden away or part of a power strip. With the use of remote control sockets, you can control the power of anything from the Pi and a web interface, enabling better control and less use of device standby modes.

Lights

A classic home automation function is controlling the lights in the house depending on the time of day or how dark it is. There are many ways you can do this: the popular method right now is Wi-Fi enabled bulbs, allowing for direct control, but you can also use the remote control sockets or use strips of LEDs that can easily light a room and are much easier to interface with. With these methods separate from the automated control, you can remotely control the lights to switch on and off as you please.

Thermostat

Technologies like Nest are becoming extremely popular, but connected thermostats have been around for a long time – longer for those with a soldering iron. While we're not going to be quite creating a thermostat that 'learns', using it for external control and monitoring is easy enough when you have the right equipment. We'll be concentrating on the monitoring part in this tutorial, using a thermistor and a bit of calibration to figure out the temperature.

Security doorbell

It's surprisingly easy to get one of these home security systems set up. Using technology created for security cameras and streaming, you can create a live feed using the Raspberry Pi that can be displayed wherever you want. This can be done quite simply on the Pi using a webcam or even the Pi camera itself, and you can even try and hook it into the doorbell if you want a really cool party trick.

Augment your home by adding your own smart infrastructure to control it, all custom-coded and controlled by your Raspberry Pi

We regularly talk about how the Raspberry Pi is the perfect little computer to have around the house doing work that requires just enough computing power to keep it running. It's not the purpose of the device, but it is just really good at it. File servers, media centres, etc – its size and flexibility make it an often surprisingly powerful tool.

And we can always go a step further. Instead of handling idle computing tasks around the house, what if we had it control the house? Through modern home automation and a bit of fancy coding, you can easily make the Raspberry Pi do a little bit more and control many aspects of your home. In this feature, we're going to run you through not only setting up the controllable lights and thermostats and such, but also how to go about controlling them. Snarky voice interface not required.

Build your own automated system

Discover how to remotely switch sockets, your garage door, control a strip of LED lights and more

Remote control sockets

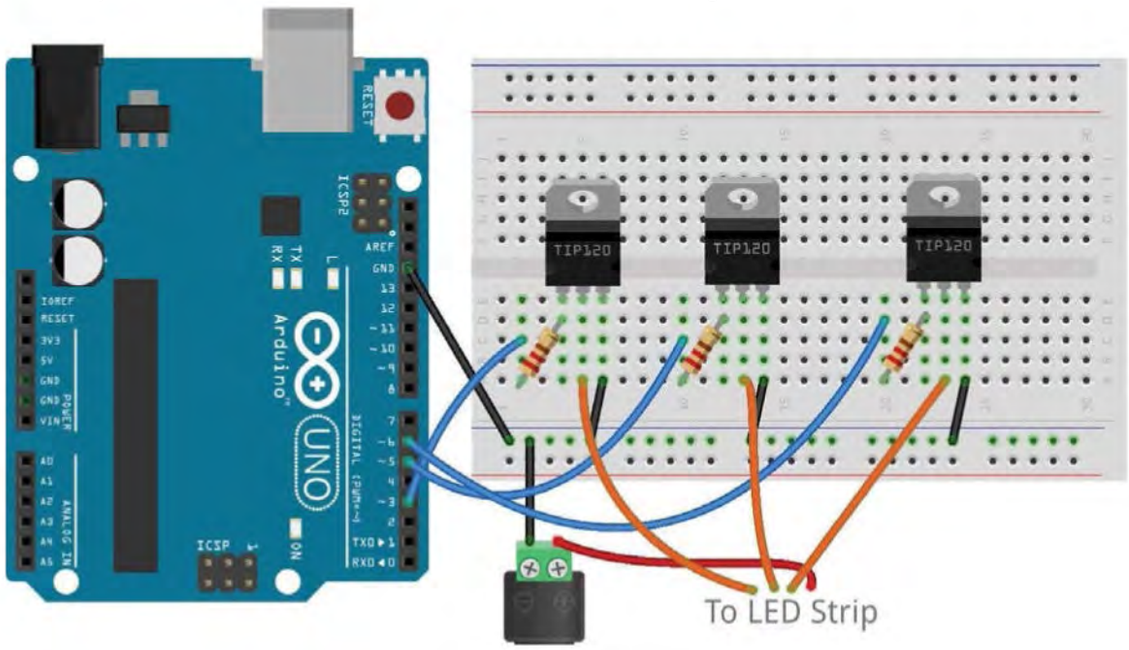
We are going to use a 433MHz receiver and transmitter module (these can be found for a couple of pounds on eBay – simply search for '433MHz' and you'll find what you're looking for) connected to an Arduino to switch a pack of remote control sockets. We used a pack of four remote control sockets from Energenie (£17 on Amazon). Remote control sockets are ideal for items such as floor-standing lamps, and anything else without an on/off switch. In our expert's case, he has an audio mixer that doesn't have an on/off switch.

Once you have the sockets set up to work with the remote, you can use the 433MHz receiver and a simple piece of Arduino software to capture the message sent by the remote so it can be sent later using the transmitter module. The modules have very simple wiring: 5V, Ground and Data. The receiver has two pins for the data line but you only need to connect the Arduino to one of them. The beauty of sniffing remote control codes is that it can be used for anything you like that works at 433MHz – remote control garage doors or light switches are likely to use 433MHz.

LED light strips

LED light strips are great. They are very easy to find on Amazon and cost about £15 for a length of five metres with a 12V power supply and remote. They have adhesive on the back so they can be stuck to a surface. Alternatively, you can just leave it on the reel as that will still give off a lot of light.

Each LED on the strip has an individual red, green and blue colour component. These colours can be set to any brightness level between 0 and 100% using pulse width modulation. This is where the signal for each colour is a square wave that is on a certain amount of the time. For example: if the Red signal had a 50% duty cycle then it would be on for 50% of the time, resulting in reduced brightness. The strip has four connectors on it: +12, Red, Green and Blue. The colour connectors are the ground pins for each colour. When the pin for a colour is connected to ground, the circuit is completed, allowing 12V to flow through the strip and to ground. We will use a high current transistor for each colour to control the connection. Note that the ground from the Arudino and power supply should be connected together on the breadboard. The 12V



Below The three separate colours used for the RGB sequence are wired manually to the LEDs

supply for the LEDs should not connect to the Arduino – only to the LED strip – or the Arduino will get damaged.

We used a TIP120 NPN transistor, which is capable of switching 5 amps – this is plenty for our application (the power supply that comes with the strip only supplies 2.5 amps, and we can switch 15 amps because there is a 5 amp transistor for each colour). This transistor would also be good for controlling the speed of a fan or other kinds of motors.

The pins of the TIP120 are Base, Collector and Emitter, from left to right. The base is connected to the PWM signal from the Arduino via a 220-ohm resistor. The collector is connected to one of the colour pins of the LED strip, and the emitter is connected to ground. Note that when passing high amounts of current, these chips get hot. Also, it is wise to use solid core wire from the emitter-to-ground and collector-to-LED strip wires because they can safely handle more current than breadboard jumper wires. You need to ensure that none of the wires connected to the strip can short, otherwise you could damage the LEDs in the strip. Our setup here is only temporary – it could be worth moving the circuit onto veroboard so that it is more stable once you have tested that it works on a breadboard.

Temperature sensor

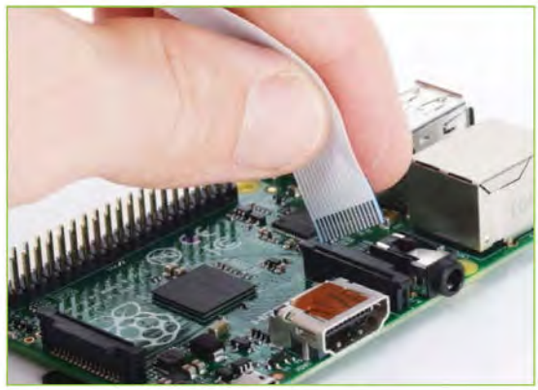
The TMP36 is a very simple IC. It has three pins: 5V supply, ground and a voltage out from 0-2V to represent temperature. This variable voltage can be read with the analogue in pins of an Arduino. The formula is: Temp in °C = [(Vout in mV) – 500] / 10, so a voltage of 0.7V would be 20°C.

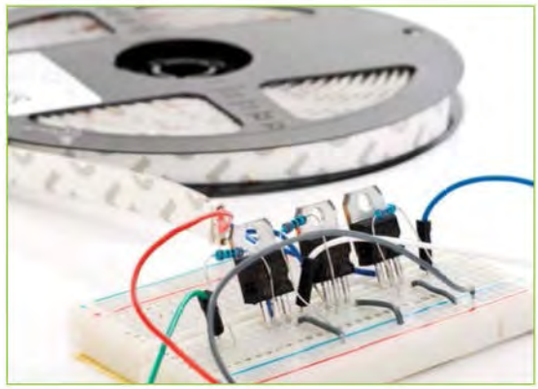
Camera

You can either use a USB webcam or Raspberry Pi camera for the video stream. The Raspberry Pi camera should be connected with the blue plastic facing the Ethernet connector, and the exposed traces of the ribbon cable facing towards the HDMI connector.

More hardware

A more involved option for home automation could be remote control curtains or remote control blinds. These are an excellent option because you can use them on a timer to wake yourself up, and also have them close whenever the sun goes down. These devices are likely to use the same 433MHz frequency that the remote control sockets use, so you could add them to the software we're going to use without much effort. If you wanted to put them on a timer you could roll your own Python script that sends the open and close commands to the Arduino (we'll discuss how this works later on) at the right times. This could be combined with Wi-Fi light bulbs so that the lights can come on as the curtains close.

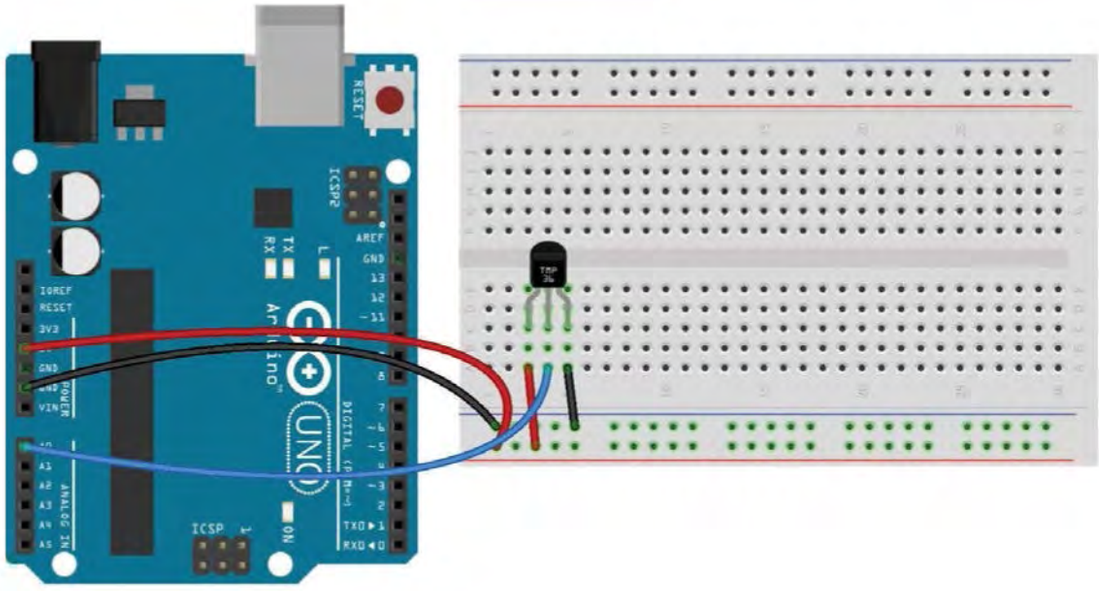




Top The Raspberry Pi camera plugs directly into the Raspbery Pi itself, not requiring extra wiring

Above Here's a better look at the colour controllers hooked up to the LED lights

Below The temperature sensor is very simply wired up, connected to positive and ground rails along with a data pin



Control your automated system

Configure your Raspberry Pi and Arduino to work in harmony for an automated home

We are going to use heimcontrol.js as the control software for our automated system. This is a home automation web interface written in Node.js that runs on the Raspberry Pi, and sends low level serial commands to an Arduino connected via USB to control various hardware connected to it. Although some of the things the Arduino does can be done with the Raspberry Pi in theory, the Raspberry Pi does not have the ability to read analogue voltages, so temperature and light sensors would not be possible in this case. Also, the Raspberry Pi only has one hardware pulse width modulation output and three are needed for the LED strip.

Raspberry Pi prep

Use the latest Raspbian image as a starting point for this project. Log into the Pi using the default username of "pi" and default password "raspberry". The first thing you'll need to do is use **sudo raspi-config** to enable the camera and resize the root filesystem so the image uses all of the available space on the SD card. Once this is done, we can update our packages to the latest version and then install some extra packages that will be required later on:

```
sudo apt-get update
sudo apt-get upgrade
```

```
sudo apt-get install libboost-dev arduino streamer
```

Next, we need to download a precompiled Node.js package and a MongoDB package (both are required by heimcontrol.js. Packages can be found at **http://liamfraser.co.uk/lud/homeautomation**, in case they go missing.

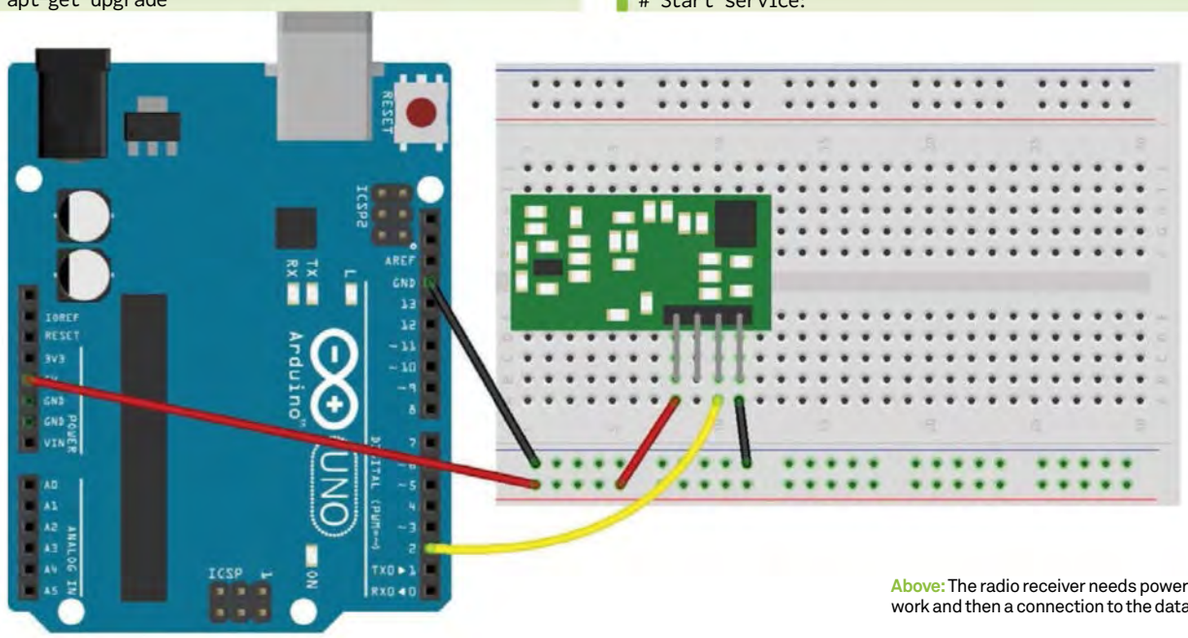
```
wget https://node-arm.herokuapp.com/node_0.10.36_armhf.deb
sudo dpkg -i node_0.10.36_armhf.deb
```

Check it has installed correctly:

```
pi@raspberrypi ~ $ node -v
v0.10.36
```

Time to install MongoDB...

```
wget https://github.com/tjanson/mongodb-armhf-deb/releases/download/v2.1.1-1/mongodb_2.1.1_armhf.deb
sudo dpkg -i mongodb_2.1.1_armhf.deb
# Start service:
```



Above: The radio receiver needs power for it to work and then a connection to the data pin

> This way of working is elegant because it allows more sensors to be added without needing to reprogram anything

```
sudo /etc/init.d/mongodb start
# Automatically start service at system startup:
sudo update-rc.d mongodb defaults
```

Now it's time to install heimcontrol.js, which our expert had to fork on GitHub to fix a couple of issues.

```
npm config set python python2.7
git clone https://github.com/liamfraser/heimcontrol.js.git
cd heimcontrol.js
npm install
```

The install process will take a while as there's quite a lot of stuff to compile. Before you can access heimcontrol.js, you will need to know the IP address of your Raspberry Pi. You can find this out using the **ip addr** command. Our address was 172.17.173.41. Run heimcontrol.js by typing:

```
node heimcontrol.js
```

Note that you have to be in the directory where you cloned the repository for this to work. This is probably /home/pi/heimcontrol.js. Heimcontrol runs on port 8080, so type the IP address of your Pi into your web browser followed by :8080 – in our case the correct URL was: **http://172.17.173.41:8080**. We have applied a patch that disables authentication by default, because it gets annoying if you have to log in every time you want to use the web interface. You can re-enable authentication by editing config/development.js inside the heimcontrol.js directory.

Now that we know heimcontrol is working, Ctrl+C out of it because we have more work to do before we can start using it. We need to load the video for the Linux camera driver for the Raspberry Pi camera so that it can be used with the streamer software we installed earlier. To do this, you need to edit /etc/modules using sudo and your favourite text editor (use nano if in doubt, so **sudo nano /etc/modules**). Add the line "bcm2835-v4l2" to the end of the file so the driver is loaded at boot. To load it instantly, run **sudo modprobe bcm2835-v4l2**.

Arduino prep and remote control scanning

We need to write some software to the Arduino called duino, which allows the ports to be controlled over serial from heimcontrol.js. This way of working is elegant because it allows more sensors to be added to the Arduino without any need to reprogram anything. We have already installed the Arduino software, so now we need to copy some libraries required by duino to the Arduino installation directory so that the software can be compiled:

```
Received 16738063 / 24bit Protocol: 1
Received 16738062 / 24bit Protocol: 1
Received 16738062 / 24bit Protocol: 1
Received 16738055 / 24bit Protocol: 1
Received 16738054 / 24bit Protocol: 1
Received 16738054 / 24bit Protocol: 1
Received 16738059 / 24bit Protocol: 1
Received 16738059 / 24bit Protocol: 1
Received 16738058 / 24bit Protocol: 1
Received 16738058 / 24bit Protocol: 1
```

Left Here are the codes that we sniffed from our remote control socket controller

```
cd /usr/share/arduino/libraries
sudo cp -r /home/pi/heimcontrol.js/node_modules/duino/src/libs/* .
cd ~
```

Before we write the duino software to the Arduino, we want to use the Arduino to sniff the messages sent by the remote for the remote control sockets. To do this, connect the 433MHz receiver module (the wider module of the two modules with four pins instead of three – see diagram to left) to the Arduino. Connect the data pin (either of the middle pins) to pin 2 of the Arduino, VCC to 5V, and GND to GND. Download the receiver software using:

```
wget https://raw.githubusercontent.com/sui77/rc-switch/master/examples/ReceiveDemo_Simple/ReceiveDemo_Simple.pde
```

... which is again mirrored over at **http://liamfraser.co.uk/lud/homeautomation/ReceiveDemo_Simple.pde**.

Now we have to start the Arduino software. If you are connecting to the Pi via SSH then you can enable X11 forwarding to your local machine by logging in with:

```
ssh -X pi@172.17.173.41
```

Alternatively, you can type **startx** to start an x session if you have a screen and keyboard connected to the Pi. Once you have logged in with your desired method, type **arduino** into a terminal to start the Arduino programming software.

Open the ReceiveDemo_Simple.pde file that you just downloaded and upload it to the Arduino. Then open the serial monitor (Tools>Serial Monitor) and press the reset button on the Arduino. By pressing each button on your remote, you can see the code to switch each socket on and off. Make a note of the codes for each button because you will need to enter them later. Our output can be seen in the top-right image.

Once this is done, we can finally write the duino software to the Arduino. This process is the same as what you've just done except the file is located at /home/pi heimcontrol.js/node_modules/duino/src/duino/duino.ino.

The software might take a minute to compile. Once it has been uploaded, you can exit the Arduino software and press the reset button on the Arduino. Now we can put everything together and start adding our sensors to heimcontrol.js.

Control software

We have used some existing software in this article because writing an entire web interface and associated control software is quite involved. Also, there is no point in reinventing the wheel. However, you can easily write some of your own software in Python that talks to the Arduino to add functionality. The duino software has a very simple interface over serial: !0113001. ! starts the message, 01 means digitalWrite, 13 is write to pin 13, and 001 is the value to write to the pin (ie set it to high). The protocol is documented at https://github.com/liamfraser/duino. You can also add some of your own plugins to heimcontrol.js. If you look at one of the existing plugins and copy how that works, it shouldn't be too difficult. It's also a good excuse to learn some JavaScript in the form of Node.js.

Set up your control interface

Now is the time to start adding our sensors and devices to heimcontrol.js

01 Start heimcontrol.js on boot

Before we start adding devices, it makes sense to start heimcontrol.js on boot. To do this, we can add a line to /etc/rc.local which is a script that gets ran at boot by the root user. The file needs to be edited with sudo and your favourite editor, for example:

```
sudo nano /etc/rc.local
```

Add the following line before the line "exit 0":

```
su pi -c "node /home/pi/heimcontrol.js/heimcontrol.js" &
```

Heimcontrol.js will be started automatically at boot from now on, but for now you can start it with **node /home/pi/heimcontrol.js/heimcontrol.js**.

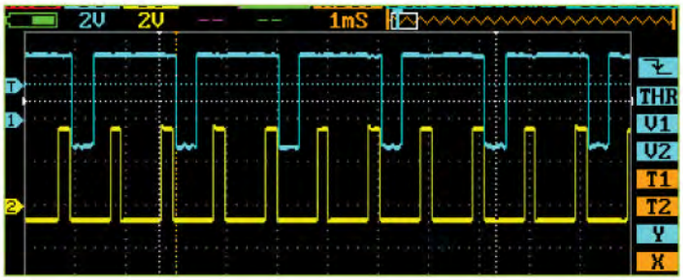
02 Add the camera feed

Go to Settings and select Webcam. Set the method to Streamer and the devices as /dev/video0. Pick an interval; we picked two seconds but you can pick any interval you like. Shorter intervals are more feasible on a Raspberry Pi 2, as it is generally more responsive. Click Save and then go back to the home page.

03 Prepare remote control socket codes

This is where you need the codes that you sniffed from the remote earlier on. The heimcontrol.js web interface takes the code as a binary string of 1s and 0s. However, the code we sniffed is in a different format so you'll need to convert it to binary using Python. Type **python2** into the terminal to open a Python interpreter. Format the integer you captured as a binary string like so:

```
>>> "{0:b}".format(16738063)
'111111110110011100001111'
```



Above The pulse width modulation signals from the Arduino. One wave is on 20% of the time, the other 80%. This controls the brightness of each colour to create any colour from the three primary colours

04 Add the remote control socket

Go to the Settings menu and go to the Arduino section. Click the Add button and set the method to RC Switch. Set the code type to binary. Give the switch a name, enter the pin that the RF transmitter is connected to (in our case, pin 2) and enter the two codes that you just worked out for the on/off buttons. Go back to the home page and test that the switch works. If it doesn't, you might need to add an antenna to the transmitter by making a loop of wire. Check everything is connected correctly.

05 Add the temperature sensor

The temperature sensor can be tricky because it needs calibrating. A multimeter is a good idea so you can accurately read the analogue voltage. Go to the Arduino settings page and add a sensor. The formula for the TMP36 is: [(Vout in mV) - 500] / 10. We read 718mV with a multimeter, which would put the temperature at 21.8°C. Experiment with the formula by seeing what the raw value of x is from your sensor, but ours ended up as: (((x+45) * (5000/1023.0)) - 500) / 10. (5000/1023 is because the Arduino has a 10-bit analogue-to-digital converter, ie 0-1023 to read a voltage up to 5V.) Note that you have to ensure you have perfectly matched brackets, otherwise the software will crash because it simply tries to eval the string you put in.

06 Add the LED strip

Finally, it's time to add the LED strip. Make sure the signal wires are connected to PWM-capable pins on the Arduino marked with a ~. We used pins 3, 5 and 6. Go to Settings>RGB Lights. Ensure you have the signal wires connected to the correct colours. We used 6 for red, 5 for green and 3 for blue. Click Save and turn on the 12V supply. Select a colour from the colour picker. Your strip should light up! Pick Red, Blue and Green to ensure everything is wired up correctly. ■

Remote access

If you want to remotely access your home automation web interface then you can use a dynamic DNS provider such as No-IP. This allows you to create a domain name that always points at your home IP address. Then if you port forward SSH on your router to your Raspberry Pi, you will be able to SSH into it from anywhere (obviously you'll want to change the default password if you go this route). From there you can port forward the heimcontrol web interface to your local machine. We have covered this in a previous Raspberry Pi File Server tutorial, which can be viewed here: bit.ly/1LacazG (instead of using port 9091, you need to use port 8080).

Expand into the future

Set up multiple systems

Now that you have heimcontrol.js set up in one room, you could extend the system by setting up multiple Raspberry Pis and have one in each room of the house – potentially a good way of using all of the older models that are gathering dust in your drawers. You could either have a master Raspberry Pi that reverse proxies multiple instances of the web interface depending on the link you give it: **/livingroom**, **/kitchen** and so on, or just simply have a bookmark for each room. If you were eager to have as few cables as possible, you could always look at getting a power-over-Ethernet injector/splitter that could send 5V power over an Ethernet cable along with the network connection.

Set up an audio system

A nice addition to home automation would be some kind of multi-room audio system. If there's already a Raspberry Pi in each room, you only need a cheap class-T audio amplifier (£20), a USB sound card for better audio quality and some bookshelf speakers to get this going. A pair of powered PC speakers would also do the trick. If you want just one set of speakers then mopidy (a music player daemon with Spotify support and a web interface) would be fine. Alternatively, you could look into setting up Squeezebox, which is multi-room audio software that was originally developed by Logitech but is now open source.

Harness FOSS

126 How to virtualise Linux

134 How open source giants are changing the world

138 Running Windows apps with Wine

144 20 LibreOffice tips & tricks

148 Bodhi Linux 3.1.0

150 Korora 22

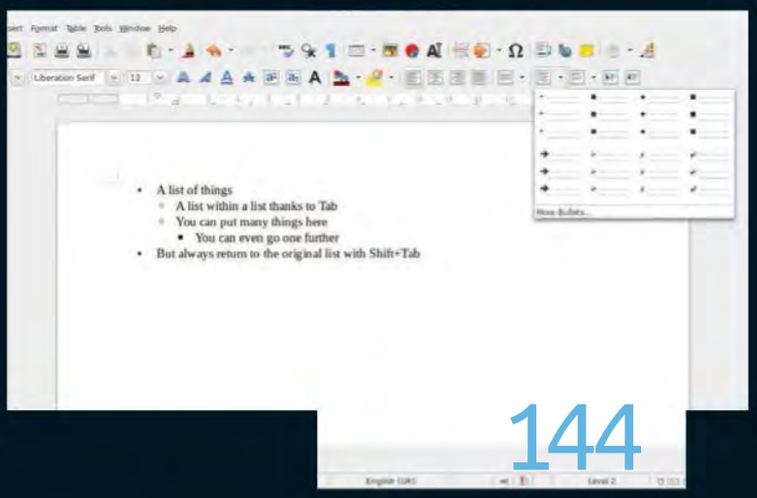
152 OSMC

154 Mageia 5

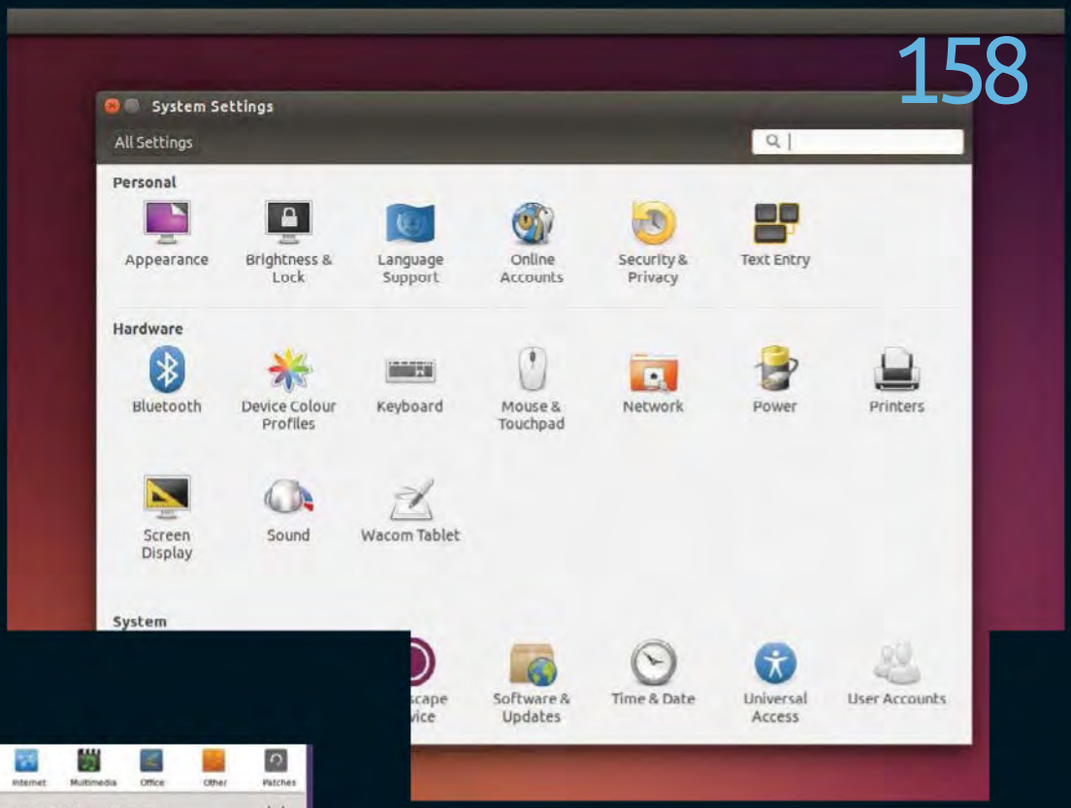
156 Solus Beta 2

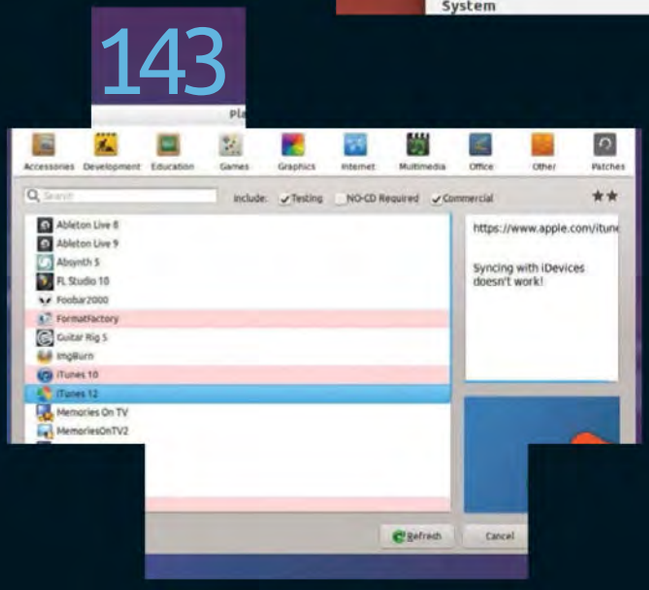
158 KDE Plasma 5

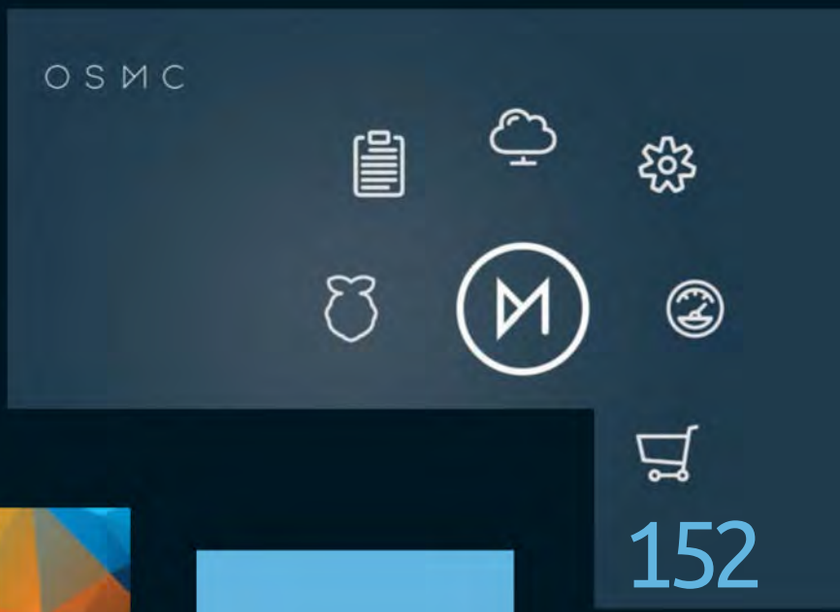
168 Code Gnome software



"Major companies are open-sourcing their most important software"

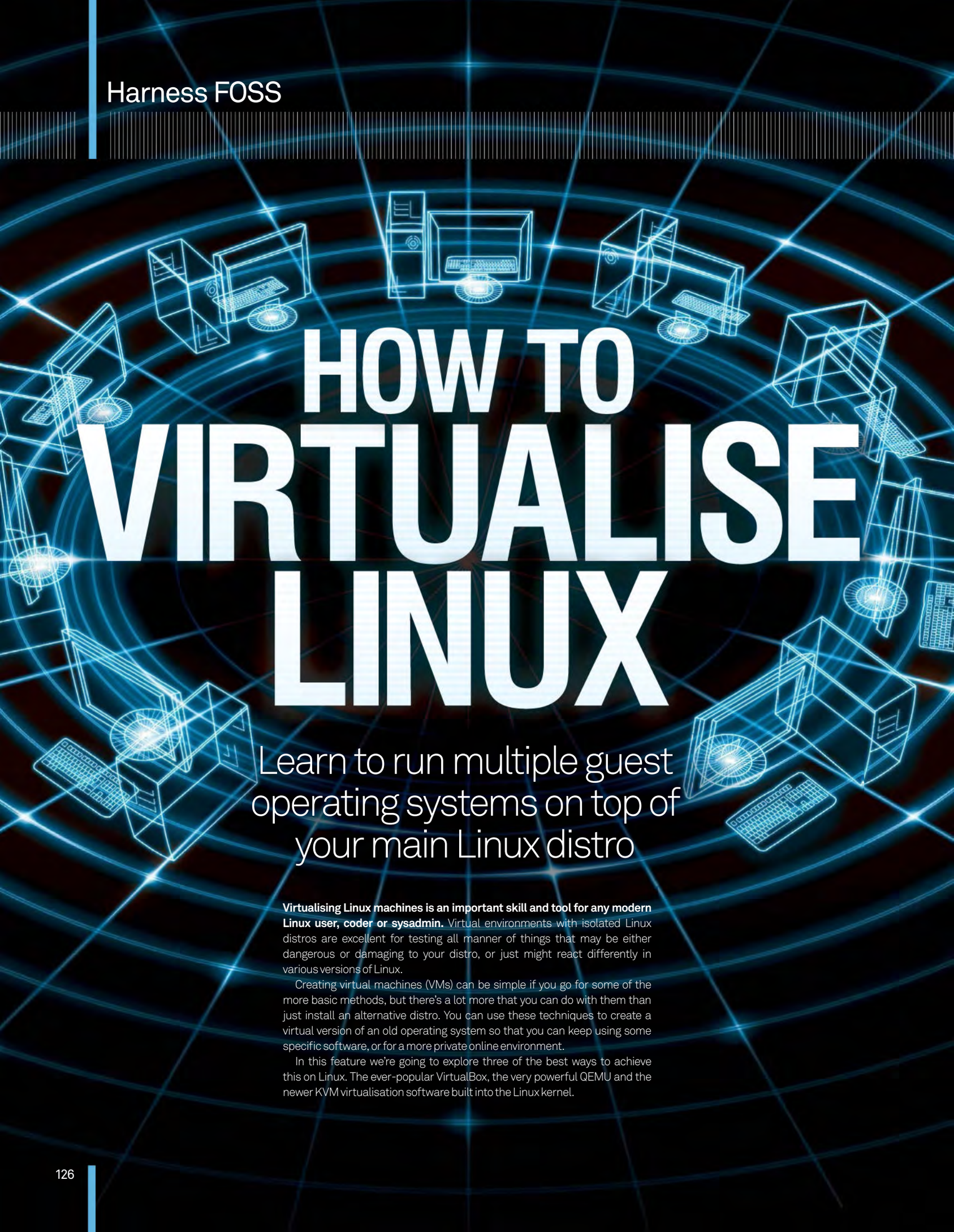

158
System Settings
All Settings
Personal
Appearance
Brightness & Lock
Language Support
Online Accounts
Security & Privacy
Text Entry
Hardware
Bluetooth
Device Colour Profiles
Keyboard
Mouse & Touchpad
Network
Power
Printers
Screen Display
Sound
Wacom Tablet
System
Software & Updates
Time & Date
Universal Access
User Accounts


143
Accessories
Development
Education
Games
Graphics
Internet
Multimedia
Office
Other
Patches
Ableton Live 8
Ableton Live 9
Absynth 5
FL Studio 10
Foobar2000
FormatFactory
Guitar Rig 5
ImgBurn
iTunes 10
iTunes 12
Memories On TV
MemoriesOnTV2
Syncing with iDevices doesn't work!
Refresh
Cancel


OSMC


150
Welcome. This is Korora
Documentation
Support
Project

152



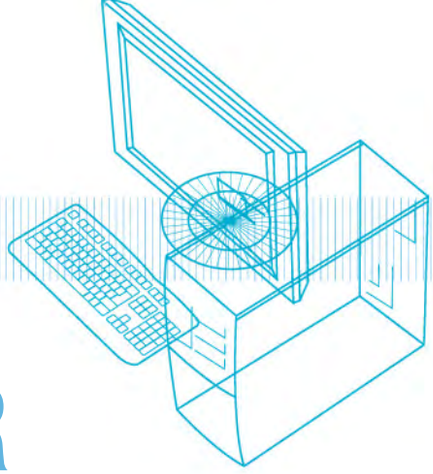
HOW TO VIRTUALISE LINUX

Learn to run multiple guest operating systems on top of your main Linux distro

Virtualising Linux machines is an important skill and tool for any modern Linux user, coder or sysadmin. Virtual environments with isolated Linux distros are excellent for testing all manner of things that may be either dangerous or damaging to your distro, or just might react differently in various versions of Linux.

Creating virtual machines (VMs) can be simple if you go for some of the more basic methods, but there's a lot more that you can do with them than just install an alternative distro. You can use these techniques to create a virtual version of an old operating system so that you can keep using some specific software, or for a more private online environment.

In this feature we're going to explore three of the best ways to achieve this on Linux. The ever-popular VirtualBox, the very powerful QEMU and the newer KVM virtualisation software built into the Linux kernel.



TOOLS FOR VIRTUALISATION

Which hypervisor best suits your needs for virtualising operating systems?

KVM

USEFUL FOR Emulation and virtualisation on servers for more dedicated VMs

KVM is the set of virtualisation modules that are built into the Linux kernel – not a keyboard and mouse switch. The benefit of having KVM built into Linux itself is that it makes it easier to turn it into a dedicated hypervisor, maybe the kind you'd use on a server, without requiring extra layers of packages and other software on top of it. It's quite quick to set up, especially via a graphical method, and has some good basic functionality between host and guest. It can be used on the command line very easily if you plan to have a headless server.

VirtualBox

USEFUL FOR Testing software, development or any kind of activity where you'd like to use a particular environment on your desktop

VirtualBox is very popular among home users. It's also very simple to use, which is probably why it's so popular. We use it all the time for testing, and it's great for creating a separate distro or versions of a distro for installing software in that's in development to see how it works under different conditions. It doesn't quite have the efficiency of KVM as it runs slightly different within the operating system, but it's much easier to use from the host system as you work on multiple things inside and outside the guest. Get it now from **bit.ly/1nlDA5d**.

QEMU

USEFUL FOR Testing and developing operating systems that run on different architectures

QEMU isn't so much virtualisation software as it is an emulator to run images off of. You can still use ISOs and run Linux distros or Windows like any of the other VM solutions in this feature, however unlike those systems you can also run distros that require ARM (such as Raspberry Pi distros), MIPS, or any other kind of processor you can think of. This is excellent for testing out software on different architectures if you don't have a relevant device handy. It can be a little tricky to use though as it relies more on a command line interface.

Optimal RAM

Giving RAM to a system depends a lot on what the host is going to be used for. Even though it's a virtual machine, it's using your host computer's actual RAM while on. If it's a dedicated server for hosting virtual machines, a lot more of your RAM resources can be given to the virtual machine. Having the host machine grind to a halt will negatively affect the performance of the VM, so you need to take into account the kind of loads the host will face. Usually, about half of your RAM is considered a good minimum, especially if your host is simply a normal desktop system.

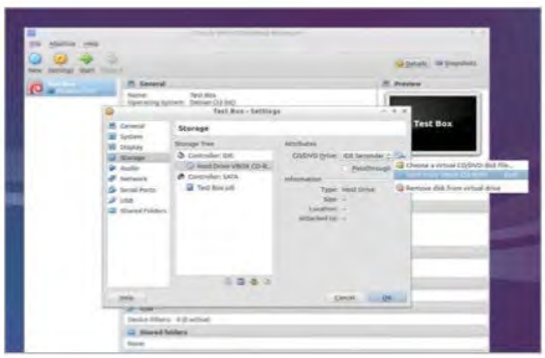
START WITH VIRTUALBOX

Learn how to make quick and very useful virtual machines with the easy-to-use VirtualBox

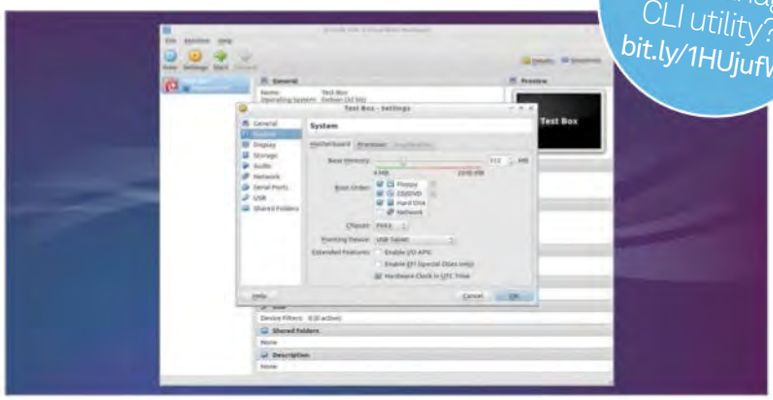
Why not try the VBoxManage CLI utility? bit.ly/1HUjufW

To create your first VirtualBox VM, open the program and click New. Set the Type and Version as close to your guest distro as possible – for example, use a Debian base for a Parsix installation. Next, you assign RAM to your VM; this will vary depending on how many simultaneous guests you plan to run, but as a general rule assign half your RAM to a guest if you plan to run just the one. Finally, you need to create a virtual hard drive. Unless you need to work with other software, like VMWare, stick with the default VDI format. Create your hard drive at a Fixed Size, rather than Dynamically Allocated, if you can; the former is slower to set up but can be faster in use. Set the size of the hard drive to a reasonable amount – plenty for the OS with spare space left over for the programs and files you'll need. About 50 GB should be more than enough.

With the hard drive set up, your VM is almost ready. Click Settings and go to the System page. As well as adjust your RAM setting, you can change the default boot order on the first tab. Click the Processor tab and you can set how many of your processor's cores to assign to the virtual machine – give half to your guest if it will be the only VM running at a time. Now head to the Display page and give yourself as much Video Memory as you are recommended (ie the green area of the slider). Once you've installed the Guest Additions (see below), enable the 3D Acceleration and 2D Video Acceleration options here for better performance. Next, head to Storage and make sure your optical drive, real or virtual (eg via Virtual CloneDrive), is connected to the guest by adding it as the IDE Secondary Master, then clicking the disc icon and selecting your drive. Sometimes, you may find that your Internet connection doesn't work properly to start with. If that's the case, head to the Network settings and change your adapter (which defaults to NAT) to Bridged instead. With everything set up, click Start to boot the machine and begin installing your distro of choice.



Above Your disk drives and any ISOs mounted on virtual drives can be selected using this drop-down menu

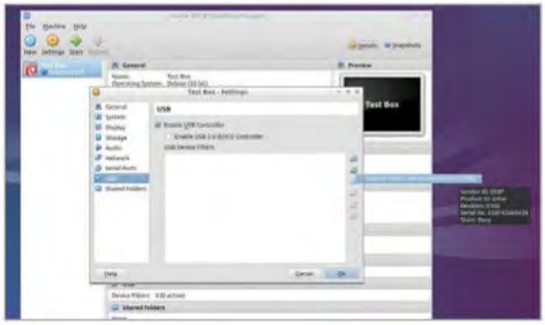


Above RAM allocations will be recommended to you during setup, but you can always go back later and adjust this in the Settings

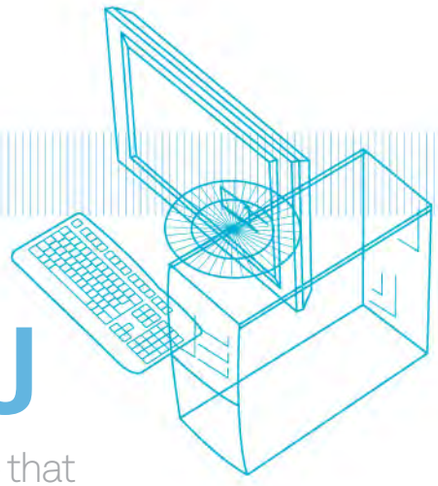
VirtualBox extensions

There's a proprietary extensions pack (**bit.ly/1nlDA5d**) that includes VirtualBox Remote Desktop support, enabling you to run a VM off one host machine while controlling the guest system from another, for when you want one powerful headless server for multiple guest systems. The pack can also enable the USB 2.0 (EHCI) controller, remote booting via the emulation of the Intel PXE boot ROM, the use of the host system's webcam, and more.

Once you're done and inside your virtualised OS, you'll probably want to install the Guest Additions. This enables you to go full-screen with your VM at a proper resolution, and it will also enable mouse pointer integration, better video support, seamless windows and shared folders between guest and host, a shared clipboard and a few more features besides. To install them, go to the main Devices menu and choose 'Insert Guest Additions CD Image'. Mount the disc, if it isn't mounted automatically, then **cd** into the folder. From there, run **./VBoxLinux Additions.run** as root. If the installation doesn't work correctly (eg you still can't display at full-screen), ensure your distro is fully up to date and then install both the **dkms** and **virtualbox-guest-dkms** packages before retrying. Finally, if you want to use USB devices inside your guest then first ensure you have the Guest Additions installed. Next, head to your VM's Settings, click USB and enable the controllers, then click the Add Filter button to the right and add your device to the list of filters. To finish, open your distro's Users & Groups settings, scroll down to vboxusers and then add your user to that group.



Above Adding your USB device to the Filters disconnects it from the host when you run the guest, freeing it for use



VIRTUALISE WITH **QEMU**

Get to grips with this more manual virtualisation software that has a lot more controllable elements and a little more power

QEMU is not so much a virtual machine software as it is an emulator, allowing you to run operating systems and Linux distros on it by emulating a processor. It doesn't just emulate your normal processor, though; it allows you to run on x86/x64 systems while giving you access to an emulated ARM, MIPS and other processor styles. This is excellent for development and testing, but it's also a bit trickier than other forms of virtualisation and doesn't allow you to quite install like you can with VirtualBox or KVM. However, you can more easily run an image of a hard drive with it.

01 Get QEMU

First we need to install QEMU on your system – it's supported by most Linux distros in their repositories, so in Ubuntu you would install it with something like:

```
$ sudo apt-get install qemu
```

For Fedora it would be **yum/dnf install**, etc. If you're a sysadmin, though, you probably want to build it from source to make sure it works best on your system – the source is available from the QEMU website (**wiki.qemu.org/Download**).

02 Create a virtual hard drive

You can create some virtual space to use while emulating the different operating systems and distros, which is useful for storing and saving data, and you can even install to it if you wish. In the terminal you can do this by using:

```
$ sudo qemu-img create vhd.img 20G
```

This creates a 20 GB virtual hard drive, although you can make it as large or as small as you want. You should probably also keep it in a working directory.

Graphical clients

While we've concentrated on how to use QEMU in the command line, there are some solutions that allow you to use QEMU with a graphical interface. Two of the more popular ones are GNOME Boxes and Red Hat's Virtual Machine Manager – we're using the latter in the KVM tutorial. Both of these can hook into standard virtual machines and allow you to find out much more information about them, as well as do a little management on the side.

03 Emulate x86

We can emulate our first operating system on x86. Make sure you have an ISO downloaded, then put it inside the qemu working directory to make launching easier. Once that's done, go back to the terminal and use:

```
$ sudo qemu-system-i386 -hda vhd.img -cdrom
[ISO].iso -m 2048 -boot d
```

... with ISO being the name of the ISO you're using. It might be a little slower than the other virtualisation solutions because it's actually emulating the system, but it will work.

04 Install on virtual hard drive

Go through the normal installation procedure for your distro (make sure you made a big enough virtual hard drive for it – the 20 GB we created should be plenty). Again, it will take a little longer than usual. Once it's installed and you do a proper shutdown, you can turn it back on again using:

```
$ sudo qemu-system-i386 -hda vhd.img -m 2048
-boot d
```

This will then boot straight from the hard drive, without needing the CD.

05 Emulate ARM

This can be used to emulate the Raspberry Pi for Raspbian, among other things, and is a good way to test out software that will run on ARM versions of other distros if you don't have quick access to a device. This requires a little change of the initial run command to something like:

```
$ sudo qemu-system-arm
```

... with the relevant image and RAM details (the **-m** option) to suit the system.

06 Emulate MIPS

Similar to emulating ARM, this allows you to test things out on MIPS hardware. It's supported on QEMU along with many other forms of CPU architecture, and for this you simply need to use the **qemu-system-mips** command to run an image or operating system. There's also support for MIPSEL and MIPS64 if you so need; just change the command to **qemu-system-mipsel** and **qemu-system-mips64**, respectively.

VIRTUALISE WITH KVM

A virtualisation tool built right into the Linux kernel, Kernel-based Virtual Machines are powerful and fairly easy to use

Also known as Kernel-based Virtual Machine, this allows for a Linux distro itself to be the hypervisor for its own virtual machines. This requires hardware virtualisation support in Intel and AMD processors (Intel VT-x and AMD-V, respectively), but it supports a lot of different Linux distros and other operating systems. It can be managed graphically so it's about as easy to use as VirtualBox while also being a little more efficient in some cases.

KVM is good for both desktops and servers used for creating virtual machines, but does require a little more setup than VirtualBox, while also being slightly easier than QEMU.

01 Test compatibility

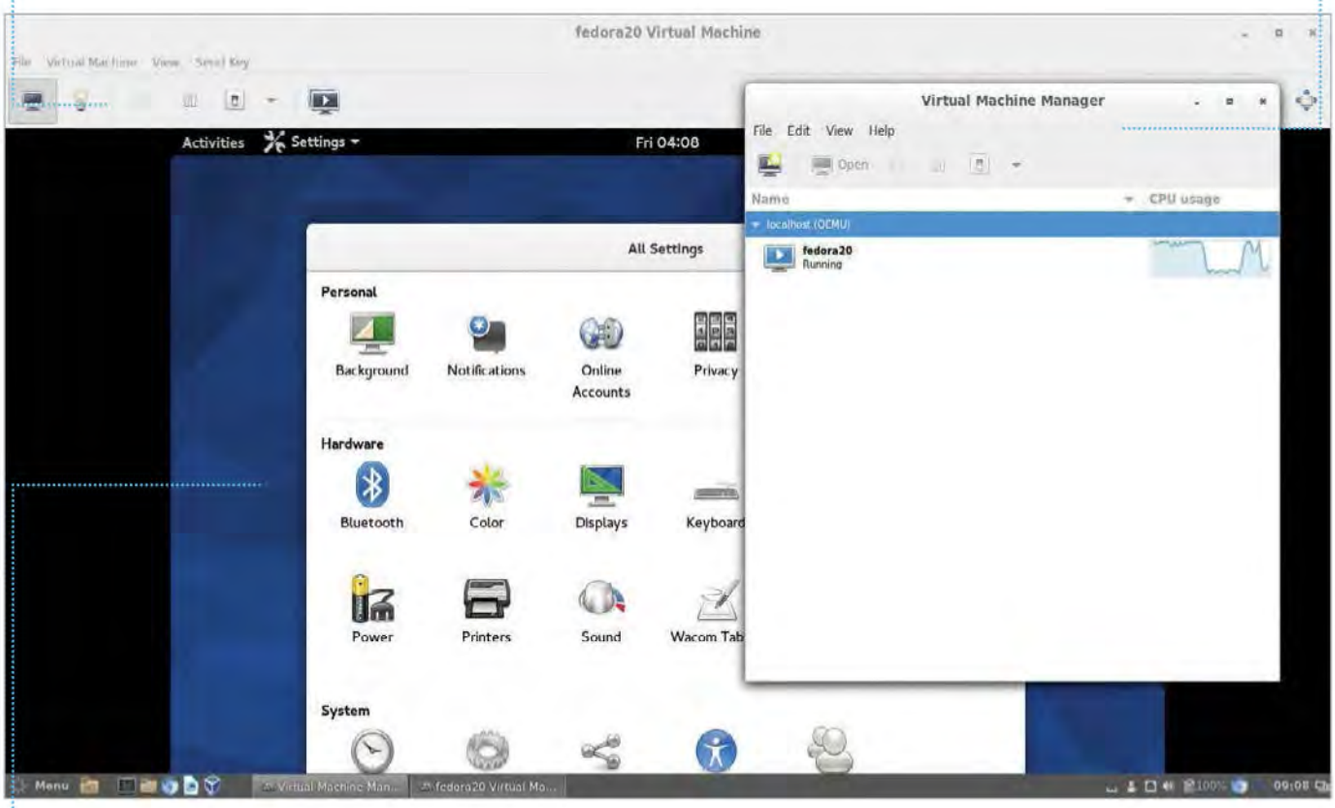
It's important to first find out whether your system can or cannot actually run virtual machines using KVM, as it requires the hardware virtualisation support to be activated. To do this, open a terminal and use the following:

```
$ egrep -c '(svm|vmx)' /proc/cpuinfo
```

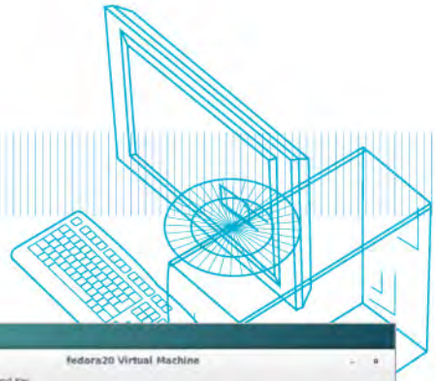
If it returns a number of '1' or higher, this means your CPU supports KVM, although you may need to activate it in the BIOS if the result of this tutorial doesn't work.

Use the virtual machine inside a powerful guest interface that enables some excellent host-to-guest controls

Create and manage multiple virtual machines that run on KVM using the VMM interface that also works with QEMU



Run virtual machines with better efficiency than some other methods, thanks to it being tied directly into the kernel



02 Install KVM

Once you're satisfied about whether your VMs will work or not, it's time to actually install KVM – although it *is* part of the kernel, it's not automatically activated all the time. To do this easily, simply open up the terminal again and then use:

```
$ sudo apt-get install qemu-kvm bridge-utils
```

This is purely the command line version, so if you want to add the graphical tools (which we're covering), you can then install them with:

```
$ sudo apt-get install libvirt-bin virt-manager
```

03 Set up permissions

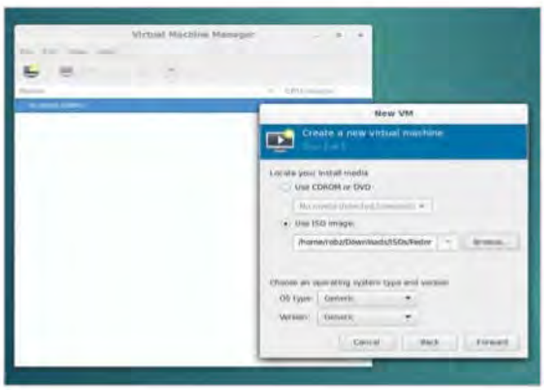
Normal users can't use VMs in KVM, so you need to elevate any non-root users you'd like to have access these VMs by adding them to groups with the correct permissions. The group in question is libvirtd, and you can add your own username to it with:

```
$ sudo adduser [username] libvirtd
```

Log out and back in again, and check to make sure it worked by using:

```
$ virsh -c qemu:///system list
```

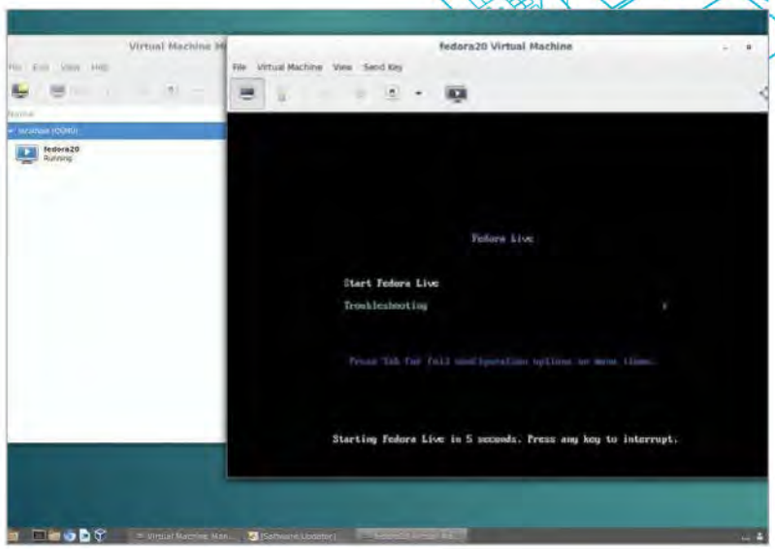
It will then list all of the working VMs, which at the moment will be none.



04 Create a virtual machine

In the menus of your distro you should be able to find the Virtual Machine Manager. Open it up and you will be greeted with the graphical interface for the system. From here you can make a bit of a shortcut to creating your virtual machines by first clicking on the 'Create a new virtual machine icon'.

We're going to be working from an ISO, so choose that, and then select the ISO that you plan to use. Give the VM a set amount of RAM (we like to go with half of what you have, but refer to our boxout on CPU usage for specifics on the matter), and an amount of storage memory.



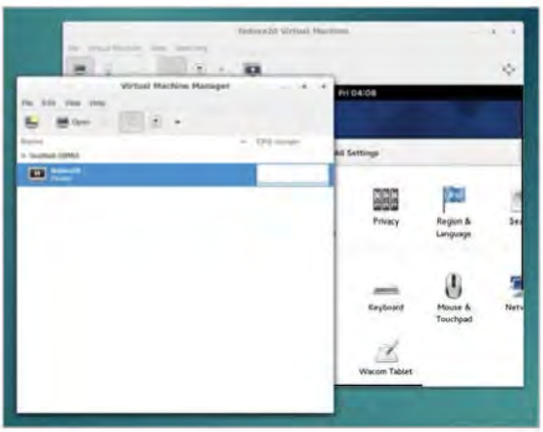
CPU usage

On all of these virtual machines we're creating, we're having to set a number of CPU cores to use from the host machine. Much like RAM usage from your host system, you can feasibly use about half the available cores without causing too much problem for the host system. If you plan to use the host while virtualising (i.e. if you're a normal user), it may be better to keep it to only one core to save some resources for yourself, although this will mean the virtual machine's performance will go down. Play about with multiple cores and figure out what suits you best.

05 Install on the virtual machine

Click Finish and you'll open the VM once it's all ready to go. Depending on your selected distro, you'll go through a different installation process – major distros will be easily able to recognise where and how to install with little trouble.

Unlike VirtualBox, you don't quite have the right-hand Control button to use as a 'host key', so you'll have to manually fullscreen and un-fullscreen it.



06 Manage virtual machines

Virtual machines can be paused, forced off, forced to reset and have standard reboot and shutdown signals sent to them. You can use the 'Send key' option in the interface while it's running to send some of the standard Ctrl+Alt shortcuts to the machine, including ones using the F keys to go between different instances.

You can't modify the virtual machines, but you can certainly create more to different specifications, or just delete them and start over. You can also 'Redirect USB device' from the 'Virtual Machine' menu on the top bar, allowing you to access external storage.

CHOOSE A **DISTRO**

Get your creative juices flowing with these ideas for Linux distributions to virtualise that can increase your productivity

SOFTWARE DEVELOPMENT

FEDORA getfedora.org

The benefit of using Fedora for development is that you always have access to the latest software to work on, and due to its use of only free and open source software it makes for a much cleaner working environment. A lot of build environments have versions available and optimised for Fedora, so you can either work in there if you really want to, or just run software in there to see how it reacts. As there's a fair amount of time between each different Fedora, you'll be able to use the virtual machine for a while without having to worry about doing a full update every six months or less. There's also very good virtualisation support in the distro to make sure that it works well in conjunction with some of the virtualisation protocols.

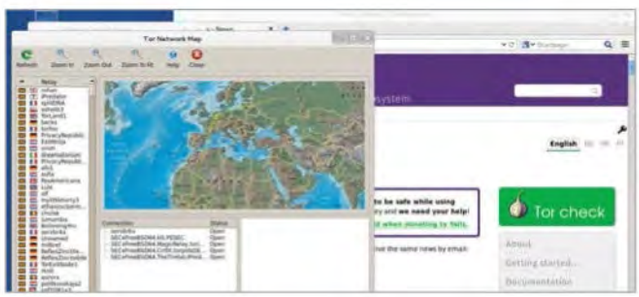
ARCH LINUX archlinux.org

This rolling Linux distro can be almost anything you want it to be. This makes it perfect for creating a testing environment requiring specific software, or a development environment that is efficient and separate from your normal system. The main difference when using Arch on a virtual machine is that you install a different graphics driver that makes better use of virtualisation (there's a specific one for VirtualBox, for example). With the rolling updates, you can always test the latest versions of software, while it's also easier to roll back and test with older bits and pieces for specific setups. You can also fairly easily create an ISO based on a particular setup, allowing you to create multiple VMs in a cloud environment with the very same setup, and very quickly as well. There are plenty of other handy features for virtualising, which you can read up on using the Arch wiki.

EXPAND YOUR OPERATING SYSTEM

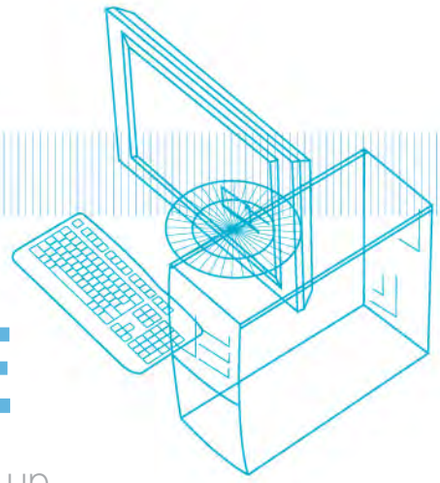
KODIBUNTU kodi.tv

If you want to turn your system into a media centre for brief periods of time, virtualising media centre software is an easy and quick way to switch into it without disturbing any of your currently running work. Probably the best media centre software available is Kodi, and the best version for virtualisation on normal systems is Kodibuntu. As the name suggests, it's a spin of Ubuntu that only has the Kodi environment, making it an excellent virtualisation base to then use Kodi properly as well. Kodi is very easy to use in general, with logically laid-out menus and great support for connecting to shares over the network or on the host. Using USB passthrough, you can even allow for remotes and other IR receivers to control Kodi. It won't be as good as a proper installation, but it will be a lot better than your normal distro.



TAILS tails.boum.org

Tails is the distro that allows you to have maximum privacy online. However, while it's generally getting more user-friendly while still remaining very private, it's not the best distro to use on a daily basis. Which is fine – it's not designed that way – however, the privacy features can be very beneficial every now and then. Virtualising Tails isn't perfect, and some aspects of its privacy won't work as well as it would by running it natively on your hardware, but for sending the odd encrypted email and quickly connecting to Tor, it still works very well. Tails does have installation abilities, but it has very little benefit over live booting from the ISO as you would normally do anyway. All the other software works just fine virtualised, so even though it's not really meant for use inside a VM, it has packages in place to allow it.



BACKUP AND RESTORE

Make sure to keep your virtual operating systems backed up and safe with these handy software tips

One of the benefits of using virtual machines is that they're easier to create backups or snapshots of specific states so that you can revert back to them in the future, or launch from the previous one while preserving the latest version. This is a good way to test software and code on different versions of the same distro without requiring several installs at once.

Backing up and snapshots are quite different in the software, so it's best to think about what you want to do. Do you want to restore an older version if something goes wrong with the VM, or if something goes wrong with the host device?

Back up and restore

As the virtual systems all exist on a virtual hard drive, it makes it a lot easier to back them up. In most cases, you can just copy the file and move it somewhere safe or secure. In something like VirtualBox, these virtual appliances also contain some of the information on how to boot the VM, which makes it easy to then restore it later without having to tweak the settings too much.

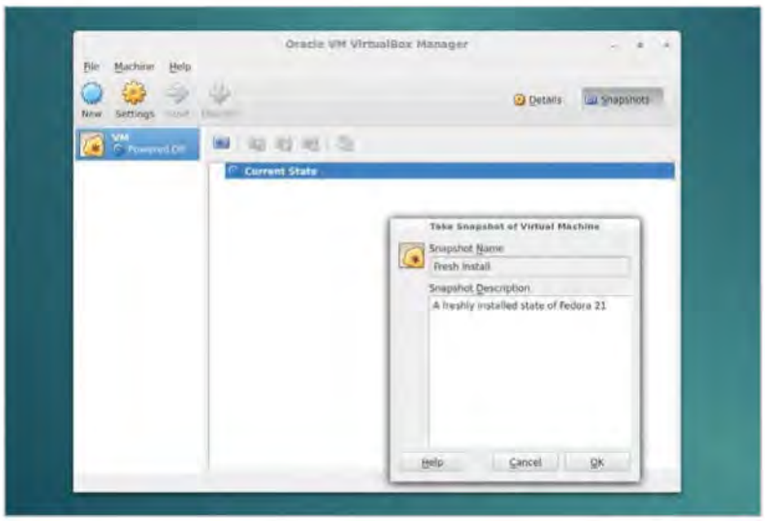
VirtualBox also has the ability to export virtual appliances into a file better suited for restoring later, which also then has all the system details of the VM. This creates an OVF (open virtualisation file) which is usable by other hypervisors and virtual machine software.

Restoring with just a virtual hard drive device is nice and easy. In VirtualBox, you can just create a new virtual machine and use an existing virtual hard drive from anywhere on your filesystem. You'll have to double-check the settings in order to make sure everything is right so that everything works as well as it should.

Snapshots

An excellent feature of VirtualBox is the ability to create a snapshot of the virtual machine at its current state. These can then be used as reference points to start working from again using older software, cloned to create a different package and upgrade path, etc.

To create a snapshot, first select your virtual machine from the manager. At the top-right of the VirtualBox interface is the Snapshots tab – from here you can press the camera button to create a snapshot of the current system. You can then select which version to start up each time, but be warned that it does eat up extra storage space.



Above Create and manage snapshots and clones using VirtualBox

> VirtualBox also has the ability to export virtual appliances into a file better suited for restoring later, which also then has all the system details of the VM

RESOURCES

Each of the different virtualisation methods will have their own little quirks and solutions for any issues you may come across. For each of these, it's best to check out the documentation or wikis of any of the described methods. VirtualBox documentation can be found on its website (**virtualbox.org/wiki/Documentation**). QEMU has a man page that's very useful, but it also has a website on which you can read other documentation (**wiki.qemu.org/Manual**). While KVM is a part of the Linux kernel, it does specifically have its own site dedicated to its usage, which also has some documentation (**linux-kvm.org/page/Documents**).

If you're still stuck after reading through the documentation, you can also check out the forums. VirtualBox has its own dedicated forum, however QEMU and KVM do not; the best place to ask about them is **linuxquestions.org**, an excellent forum filled with smart people who can help you fix the issues you're likely to encounter. Also, for more news on virtualisation on a more developer and sysadmin level, you can also check out **virtualization.info**, which reports any of the important updates in the VM community, and even our website for a few virtualisation tutorials.

HOW OPEN SOURCE GIANTS ARE CHANGING THE WORLD

Major tech companies are open-sourcing their most important software – here are the biggest projects

There have been talks about open source thinking going mainstream for quite some time now, but talking doesn't always mean something. To really check if there has been growth, innovation and adoption of open source software, we need to take a close look at big companies that dictate the ways web and related technologies are shaped.

But, there are few questions – why should large corporations bother with open source? What is the benefit for them? Isn't open source already free? The first thing to understand here is the way technological innovation works – it's all about compatibility and interconnectivity. Today it's almost impossible to create something new that is completely, truly standalone. At some point in time, software has to talk to some other software to get things done. So, collaboration is the key for the upcoming technologies, and open source does it best. There may be other specific reasons too, depending on the company in question. For example, a social media company like Facebook faces no competitive disadvantage by releasing its stack; rather, this accelerates the innovation around the core stack and helps improve it. For software vendors like Microsoft, the issue is more about being open and inclusive as it continuously faces tough competition from Linux. If Microsoft still uses its own OS on the cloud, it simply can't continue because a huge part of the cloud is already Linux-based. So, open source support has become more of a necessity for the company.

A study of how well these companies treat open source software can give a good idea of things to come. In this article, we will take a look at some of the biggest giants in the tech world, their contributions to open source and how they engage with the open source community.

GOOGLE

We know about Android and Chrome since we use them daily, but Google has more to offer

Google is one of the biggest contributors to open source software. Some of its most sought-after products like Android, Chrome etc are open source, but more importantly, Google consistently encourages more people to contribute to the open source cause via its initiatives like Google Summer of Code, Google Code-in etc. If you are not aware, Google Summer of Code (GSoC) is a global program that offers student developers, aged 18 and above, stipends to write code for various open source projects. GSoC has run for three months every year since 2005 and claims to have brought together 8,500 successful student participants from 101 countries. This year's GSoC is over, but if you are a student interested in open source then you should definitely try it out in the next year.

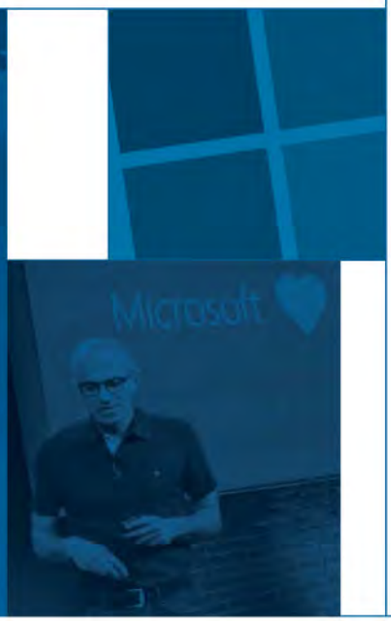
Google Code-in is another similar program for students in the age group 13-17, where they are given smaller tasks to get them introduced to the concept of open source. Google Code-in has been running since 2009, and this year's Code-in program has not started yet – if you or someone you know is a good fit for the program, keep an eye on the Google Code-in website.

Regarding open source software released by Google, one piece, Kubernetes, has recently made waves with the 1.0 release and the news of Google ceding its control and donating it to the Cloud Native Computing Foundation. Kubernetes, launched in February 2014, is now production-ready. Other notable projects are Tesseract, one of the most accurate open source OCR engines currently available with support for 39 languages, and Golang, one of the best languages for concurrency and server-based applications.

MICROSOFT

TypeScript

TypeScript, developed and maintained by Microsoft, is supposed to be a JavaScript superset. In simple terms, it lets you do far more things than JavaScript and, when you compile it, you get JavaScript code that runs on any browser. So, what exactly are the extra things you can do with TypeScript? First up, you can create enums in TypeScript, enabling you to create friendly names for number sets. Another important deviation TypeScript allows is the ability to use an object-oriented, class-based approach in programming. This means that instead of using functions as basic means of building reusable code, you can use the object-oriented approach for more modular code. This was just the tip of the iceberg though, as you can now dig deep on the official TypeScript language website for more.



Microsoft doesn't typically crop up when we talk of open source projects, but things are changing

With the new CEO openly claiming, "Microsoft loves Linux", it is clear that Microsoft has slowly embraced the open source culture. As per the Openness website, 20% of the virtual machines on Azure (Microsoft's cloud platform) are Linux and the VM depot has more than 1,000 Linux images. Azure also has first-class support for open source languages like Java, Python, Ruby, PHP and even Node.js.

On the software front, Microsoft open-sourced one of its key offerings – the .NET Core stack – and is porting it to both Linux and OS X. Open-sourced code includes the .NET Common Language Runtime (CLR), the just-in-time compiler, garbage collector and Base Class libraries. This means that you can now develop and build your code on other platforms like Linux, OS X etc while still using ASP.NET.

Microsoft also backs two of the most well-known Linux container-based programs: Kubernetes and Docker. So yes, the future looks bright for open source in Microsoft.

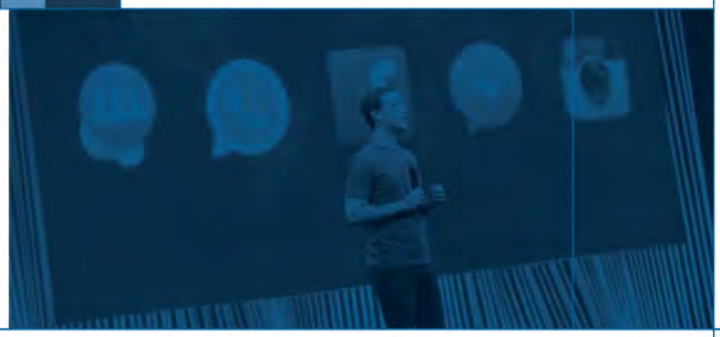
FACEBOOK

Facebook never bats an eyelid when it comes to acknowledging its open source roots

When Mark Zuckerberg wrote the first line of code for Facebook, he did it in PHP and used MySQL as the database. Facebook may have moved miles ahead, but one thing has not changed – the company's support for open source culture. Or rather, it has increased with time. Note that Facebook doesn't just share some tools to push its website – instead it open-sources its own stack (virtual machines, languages et al). Head over to its official GitHub page and you will find 274 public repos. It's very difficult to choose just a few of them for this space, but here are some of the more interesting initiatives.

HHVM, aka the Hip Hop Virtual Machine, is one of the most talked-about open source projects from Facebook. It is designed to execute programs written in PHP and Hack (an open-sourced programming language by Facebook). The major advantage it has is the just-in-time compilation technique that enables for optimisations otherwise not possible in statically compiled code. Though HHVM has a few minor incompatibilities, it can run most of your PHP code straight out of the box.

Fb-flo is another offering out of the 274 from Facebook that lets you edit live code right out of your browser. Fb-flo is made up of server and client components – the server watches the source files for changes, which are then mapped into resources and sent to the client to update the running app. The client is currently a Chrome extension that recompiles the changes sent by the server and so you have your code live-patched.



INTEL

Build world-class hybrid apps with Crosswalk

As more and more new devices are launched every year, there is an increase in the overall number of types of devices. This means more devices for app developers to support. If you are an app developer, you already know how tough this is. Crosswalk from Intel is an open source HTML application runtime for iOS, Android, Tizen and Linux. Crosswalk lets you use standard HTML5, CSS3 and JavaScript, features and APIs (that you'd use when building a website) to create your app. You can also add custom features provided by the platform you are developing for – this means, the best of both worlds.

Intel mainly comes to mind as a hardware company, but it also does a heck of lot of work on open source software

Intel is one of the core members of the Tizen technical steering group; platinum sponsor for the OpenStack foundation, number one contributor to Linux Graphics, number three contributor to Android the world over. This list could continue on for a few pages – check out the impressive project listings at **https://01.org/projects** – but apart from being open source, all these contributions are in the software field.

Hardware and software development go hand in hand, and Intel knows this very well. With contributions to strategic areas like mobile operating systems, cloud-computing software platforms and so on, Intel makes sure that it keeps up with and also shapes the software development in order to enable their own core business of hardware development.

Intel has also taken the top spot in 2015's 'Who writes Linux?' report, once again becoming the biggest kernel contributor.

ADOBE

On the open source front, Adobe has more than 200 projects, taking it near to Facebook

The average user will most likely associate Adobe with PDFs, Flash and Photoshop, but there is a lot more to Adobe than that. Though primarily the tools from Adobe are all proprietary, things have changed recently with a lot more effort being pushed towards open source software. To be precise, currently Adobe boasts 221 public repos in 79 languages on its GitHub page – that is quite a lot. Let's take a look at some of its more interesting software.

Brackets is a modern text editor you can use for web design. As expected, it integrates very well with other Adobe products like Photoshop, but you can use it as a standalone editor and you will not be disappointed. Brackets is very similar to Atom (another text editor from GitHub) in its look and feel. Also, it follows an open architecture enabling themes and plug-ins to be installed. Another interesting feature is the built-in live preview, so you don't have to go to your browser after making changes to the source code.

If you fancy writing an app for your idea, PhoneGap is your friend. PhoneGap is a development framework that enables you to create new apps with general web technologies like HTML5, CSS and JavaScript using standard APIs – but for any platform. Though these apps will not be native for the platforms, you can get a good idea of how things will work and the way forward in terms of development.

Snap.svg is an open source JavaScript library from Adobe that lets you create interactive, resolution-independent vector graphics for your next web project. It supports all the newest SVG features, like masking, clipping and gradients, and it supports SVGs from Illustrator, Inkscape and Sketch.

IBM

Node-RED – your new visual tool for IoT

The tech industry is always on the lookout for the next big thing, and the Internet of Things is surely one of them. Node-RED is a visual tool for wiring the Internet of Things. Created by IBM's Emerging Technology team, this tool provides a browser-based flow editor that makes it easy to wire together flows using the wide-range nodes in the palette. Flows can then be deployed to the runtime in a single click. Built on Node.js, Node-RED works on a selection of predefined nodes. These nodes can be social, like Twitter, email, IRC etc; network nodes, like HTTP, TCP, UDP etc; hardware nodes like Raspberry Pi, BeagleBone Black; and storage nodes like Filesystem, MongoDB, MySQL etc.

IBM is more than 100 years old and still going strong – open source is one of the reasons

Started in 1911 as CTR (Computing Tabulating Recording company), after the merger of Tabulating Machine Company and the International Time Recording Company, IBM got its current name in 1924. IBM is one the biggest employers worldwide, and it also has a long history of open source contribution and usage. For example, IBM created the famous IDE Eclipse in 2001. Another push came in the same year when IBM announced a $1 billion commitment towards the development of Linux. IBM has also been active in the Apache Software Foundation, contributing to several of its projects like Apache HTTPD, OpenJPA, Tomcat and more. Outside of the ASF, IBM also contributes to OpenStack and CloudFoundry.

IBM has created a strategy it refers to as Open Source Plus; the company aims to contribute and innovate in sync with the open source community, and then use the open source solutions within its commercial software.

RUNNING WINDOWS APPS WITH WINE

Wine gets stubborn Windows-only apps running comfortably on Linux – so no more need to dual boot!

As your day-to-day computer use – from games, through office and collaboration apps, to social media – moves to the web browser, relying on cross-platform, free and open source technologies, there's a fairly level playing field between Linux and its proprietary rivals, most particularly Microsoft Windows.

Before free software advocates rejoice too much, there remains one road block for many users looking to transition away from the spyware-ridden recent versions of Windows: a stubborn collection of legacy or proprietary software available only for Microsoft's platform.

Whether you're an office user tied to macros written for MS Office, a worker dependent upon custom proprietary software for their day-to-day work, or a gamer who simply must have their fix, a serious move to Linux will only happen if you can happily run your critical software on your new favourite OS. Enter Wine.

WINE stands for WINdows Emulator – or for Wine Is Not an Emulator; the 'not' version making it clear that Wine emulates no processor code, but instead replaces system calls to libraries and the Windows kernel with its own processes. The Wine team have done a fine job of keeping up with all of Windows' changes and developments, with the winetricks program handling workarounds and tweaks for programs that won't run on default settings.

In this, they are complemented by PlayOnLinux, which makes installing Windows programs on versions of Wine easier, keeping them partitioned in different containers.

INSTALL WINE AND PLAYONLINUX

With many versions available, we show you which to choose and how to install successfully

Your distribution makes installing most software very easy: you can **apt-get install wine winetricks** (or **yum install** from Fedora, CentOS and so on) and you'll have a recent version. A recent stable version, that is: try **wine --version** and you should get an answer of 1.6 or something thereabouts. This is fine for running long-since-released software, like that little proprietary app that someone wrote for your office back before Windows XP had appeared.

Beta

For recent software, development versions are available. The 1.7.x versions of Wine are beta versions; they're not guaranteed to work as well, but contain the latest code to get recent games, iTunes versions and more, working better than the last stable release of Wine.

The **winehq.org** website contains helpful instructions on how to get Wine onto most GNU/Linux distributions. For Ubuntu, adding the Wine repository is the first step: **sudo add-apt-repository ppa:ubuntu-wine/ppa**. After this, let Ubuntu update its database: **sudo apt-get update**. Now install the latest version with **sudo apt-get install wine1.7 winetricks**.

Winecfg will also be installed, consequently enabling a GUI-based selection of libraries, screen size, Windows versions, media drives and even integration between Windows folders and those on your OS.

Winetricks is a handy set of helper scripts to get library dependencies (DLLs) and fonts. It's also useful for helping run older versions of Internet Explorer and of course for testing page rendering. For when Winetricks is inadequate, PlayOnLinux should get some of the fussier Windows programs installed on Linux.

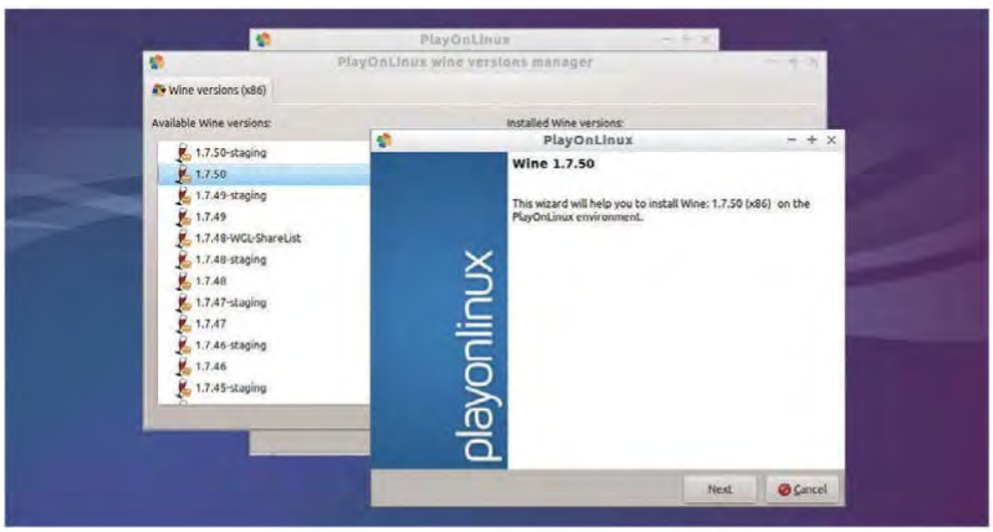
PlayOnLinux

Installing PlayOnLinux varies considerably between different distributions. On the current Ubuntu LTS, running the following steps:

```
wget -q "http://deb.playonlinux.com/public.gpg"
 -O- | sudo apt-key add -
sudo wget http://deb.playonlinux.com/
 playonlinux_trusty.list -O /etc/apt/sources.
 list.d/playonlinux.list
sudo apt-get update
```

...simply adds the PlayOnLinux repository to apt's source list. You can now install PlayOnLinux through apt, or through Ubuntu's other package management tools: **sudo apt-get install playonlinux**.

From PlayOnLinux's tools menu, choose Manage Wine Versions. From here, you can select different versions of Wine to download and install. As a result, if somebody reports definite success getting a particular game or other piece of software working with, say, Wine 1.7.42, then you can let PlayOnLinux use that one (see the image below). On a 64-bit computer, the 32-bit wine versions are on a separate tab.



Left Installing Wine through PlayOnLinux gives you complete version control

Customise for each program

Wine is capable of emulating every version of Windows, although releases before Windows 3.1 are now deprecated (do you desperately need to run anything that old?). Often it's straightforward, but for apps that need their Windows environment just so, we can get Wine to play along with winecfg settings. Using the environmental variable WINEPREFIX, we can enable custom directories for different apps. You can even run two programs at the same time with different WINEPREFIXes; for example, to model client-server software on the same machine. For 32-bit installs on a 64-bit machine, for instance, call:

```
WINEARCH=win32 WINEPREFIX=~/my-winedir winecfg
```

...and the config dialog will come up on-screen ready to tweak for your installation, which you will do with:

```
WINEARCH=win32 WINEPREFIX=~/my-winedir /path-to/setup.exe
```

To avoid having to type the same prefixes each time, use the following export command:

```
export WINEARCH=win32
```

PlayOnLinux lets you do much of this through its options, bringing up the winecfg GUI and checkboxes of libraries to install, as well as enabling the choice of a specific 32- or 64-bit Wine version for each installation.

PHOTOSHOP ON LINUX

Adobe's recent versions of Photoshop can run under Wine as PlayOnLinux takes care of the complications

Remove old Wine stains

Wine has an uninstaller GUI and **wine uninstaller <appname>** for the command line. Deleting **~/.wine/** will remove all Wine-installed apps, unless you've used PlayOnLinux. Within PlayOnLinux, separate containers make managing installations easy, and each installation lives in **~/PlayOnLinux's virtual drives** (yep, a directory name with spaces and an apostrophe). Each application has a directory hierarchy of the Windows files needed. Within PlayOnLinux, selecting an app and clicking uninstall takes care of it. You can even create a directory with the same name during a PlayOnLinux install and let it overwrite the old one. To remove Wine, while your package manager (or s**udo apt-get --purge remove wine)** takes care of the software, your desktop will retain traces, such as "Open with..." options in menus – see the Wine FAQ for help.

Of all the applications keeping people from migrating away from proprietary platforms – and discounting games for now, as few businesses depend upon their ability to run *Grand Theft Auto* in the office – Photoshop is the strongest anchor to a Windows system for many people.

GNU Image Manipulation Program (GIMP) does more or less the same thing as Photoshop and even has much better batch processing for many jobs, but the UI is different. And however compatible GIMP developers make their offering with Photoshop in terms of functionality and the file formats handled, many users have invested thousands of hours in learning their craft on Adobe's software. They're not keen to move, so let us bring Photoshop and Linux together.

Download the multi-gigabyte zipped file of the Photoshop version you need. We went for 'Adobe Photoshop CS6 Design and Web Premium', with the DesignWebPremium_CS6_LS16.7z file weighing in at 4.5GB. Then download the installer program: in our case, DesignWebPremium_CS6_LS16.exe.

Before starting, do **winetricks atmlib** or you won't be able to open PSD files! From PlayOnLinux's installation page, select the graphics tab and then 'Adobe Photoshop CS6'. You'll be told it also works with the CC (Creative Cloud) version of Photoshop.

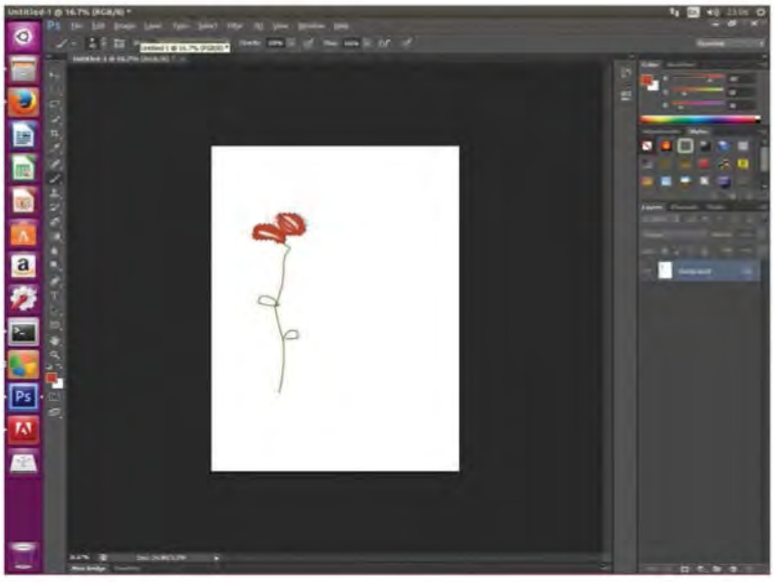
PlayOnLinux selects the appropriate version of Wine – downloading it if it's not already on the system – and all libraries/DLLs, as the settings optimised for Photoshop CS6 are already installed. A Wizard (see the bottom-left image) walks you through the process.

After clicking through the usual End User Licence Agreement (EULA, possibly the most unread category of documents of our time, but a novel experience on a Linux desktop), unzip and installation begins. You can select a 30-day trial if you have no install key handy. There's a lot of downloading too, including the ubiquitous fetching of Microsoft fonts. Unless you have a lot of RAM, super-fast Internet and a quick SSD hard drive, you may want to put the kettle on at this point. Or even go out for lunch.

Back from lunch? Good. Adobe's claim that you'll get "blazingly fast performance" with the Mercury Graphics Engine could be behind CS6's seeming inability to install properly with Wine on our test machine with an Intel graphics chipset. Our NVIDIA-equipped box (with default Ubuntu installed and no driver tweaking) worked just fine.

Not being of a particularly artistic nature (as shown below), all that can be reported of the quality here is that Photoshop seems to work quite well, with a gold rating for compatibility in the Wine database.

Above PlayOnLinux Installation Wizards are about as straightforward as they come



Right If you get runtime errors, try setting PlayOnLinux too Windows 7 for your version of Photoshop

GAMES WITHOUT FRONTIERS

From this year's latest to timeless classics, Wine helps gamers preserve their collections on Linux

Modern games

In the past, a lot of effort went into getting the Windows version of Steam on Linux. This was in order to bring the last few non-Linux games, such as *Terraria*, to the Linux desktop. Now that almost every popular Steam game is on the Linux version, or on its way there, the gamer's biggest need from Wine is to get games like *FIFA 15* working. To get that and other EA games such as *Sims 4* on your Windows-imitating PC, the first step is to install EA's downloader – Origin.

Getting the OriginThinSetup.exe file from **origin.com/download** is the first (and easiest) step. In fact, installing from the non-thin predecessor to OriginThinSetup.exe was much easier on Wine, but EA doesn't make that available any more and you really don't want to be trying an old copy from a dodgy download site.

Precise instructions will vary with your distro, Wine version, what EA has done to the installer and depending on the direction in which the wind is blowing.

The best way of getting this installed actually involves going through the motions of installing it, after which the install fails. You then go to a fresh install on a Windows computer to grab the folder **C:\Program Files (x86)\Origin**, install this over the top of your Origin folder in your Wine or PlayOnLinux directory, then run **Origin.exe** through Wine. This one worked well, enabling us to then install *FIFA 15*, but it's unsatisfactory as you may not have access to a fresh Windows installation of Origin.

On a good day, you may have better luck with the dotnet54 library, directx9, various fonts, msxml3 and lib32-libxml2 than we had. But with the older **OriginSetup.exe**, it can definitely work!

Classic games

Not everyone wants to run the latest releases, with their ever-hungrier demands for faster systems. After all, we have consoles for that – and Steam's efforts have brought a richness to the choice of games for Linux users. But permit a little nostalgic recollection of the titles of yesteryear and the lure of *Railroad Tycoon*, *Monkey Island* or *Age of Empires* could see you hunting out dusty CD-ROMs from the attic and seeing how well Wine works with software intended for Windows 98 or even earlier.

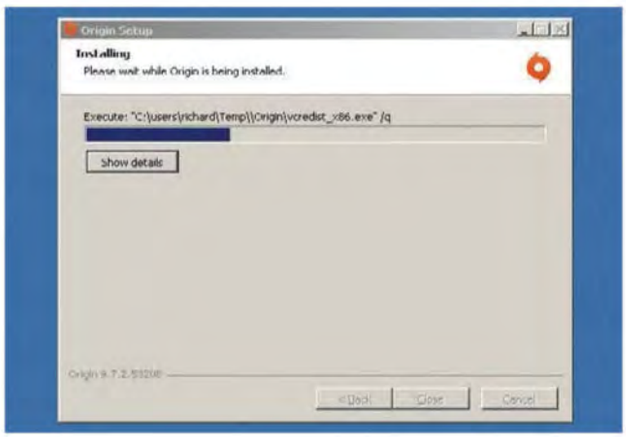
We started out our test with the original *Dungeon Keeper*. Advertised for Windows 95 or DOS, it actually runs on DOS and Wine turned out not to be necessary – install DOSBOX for that one if you still have a copy. Moving through our 90s discs, we fished out *Baldur's Gate*, a sophisticated and absorbing implementation of classic *Dungeons & Dragons* that you probably shouldn't install without first emptying your calendar.

Right-clicking the SETUP.EXE file on the CD-ROM brings up the option to open with Wine, but simply clicking it will work from most distros (or run it from the command line). In winecfg, we set Windows 95 as the OS and left graphics at the default values. As the installation rolled by, options such as "200MHz or faster processor" reminded us just how far back in time we were travelling.

Everything went flawlessly, as you can see below, and *Baldur's Gate* duly appeared in the Wine menu. This is also a game that you could do on the stable version of Wine. We don't have time to tell you more, however, as there's treasure waiting in that dungeon...

Bottom left Wine works with most newer games, but EA's Origin platform can be capricious

Bottom right Classic Windows games can easily be resurrected and added to the party



RESCUE ITUNES CONTENT

If you really need iTunes then forget the confusing online advice and follow PlayOnLinux's Wizard

Recent bloat to iTunes has seen it grow from the relatively light, focused, powerful music library app of the early days, to a tottering mass of scarcely needed functionality that is the multimedia equivalent of MS Office. There are plenty of Linux alternatives to iTunes too, although almost all have their drawbacks. Nevertheless, for users with purchased media trapped inside Apple's walled garden, iTunes has its advantages and many Linux users need access to it without having to dual boot.

Unfortunately, running iTunes on Wine is not always easy and many go to the extent of keeping a Virtual Machine (VM) on standby, running a recent OS X or Windows version just to run iTunes.

There is a mass of contradictory information only a web search away on installing iTunes with Wine. We tried following some, adding in various libraries, such as libmsxml3, riched20 and dotnet40 through winetricks and Winecfg, then through PlayOnLinux's library loader. The best we got was successfully installing a greyed-out box, although some menus were accessible. Even starting again after **winetricks gdilib**, which downloads 600MB of graphics library, didn't fix it.

Then we forgot about the postings online saying what we should do. We just sat back and trusted PlayOnLinux. After clicking install, instead of selecting non-listed software, we went to the MultiMedia menu and clicked iTunes 12 (see the bottom-left image). We also downloaded the 32-bit installer from Apple, even for our 64-bit test rigs.

PlayOnLinux warned us about what wouldn't work: syncing with iDevices, although there are plenty of native Linux apps that will talk to Apple music hardware (on a good day at least). Then, as on our Photoshop installation, PlayOnLinux's Wizard did all of the hard work, including selecting Wine 1.7.48 as the one known to work by that particular installer's creator, and wine_gecko for rendering the web interface to Apple's iTunes store. Then it fetched various other service packs and libraries. iTunes started successfully on all three test machines.

As reported, plugging in an iPod or iPhone didn't work and managing a music collection was hit or miss, as very large disks of multimedia files could crash the program. However, we could visit the iTunes Store to spend our hard-earned cash and, best of all, iTunes has a vast selection of specialist Internet radio stations. We spent a happy evening listening to bebop on **calmradio.com** – available on iTunes without a membership login – and all that jazz. ■

Registry surgery

Wine can call its own regedit for you, functionally similar to the Windows software of the same name, if editing the Windows registry is really necessary. One of the advantages of running under Wine is that you won't make your system unusable if you go badly wrong, and a backup of **~/.wine** or the PlayOnLinux equivalent location(s) can mean not even losing the emulated installation either. The files edited live in **$WINEPREFIX/*.REG**, but should not be directly edited as special encoding is used to store keys. From regedit, **HKEY_CURRENT_USER\Software\Wine** is likely to be your main focus. Under this, for example, at **MSHTML\GeckoUrl**, is the location of Wine_Gecko, which is downloaded each time you create a new **$WINEPREFIX**. Changing the registry entry to somewhere local, like **file://Z:\path\to\wine_gecko.cab**, means no unnecessary downloads. The most useful keys to modify are listed with useful explanations at: **wiki.winehq.org/UsefulRegistryKeys**.

Above Always check the PlayOnLinux menus to see if your desired software is already listed

Right iTunes will struggle to manage large music collections, but at least you can get to your Store-bought goods

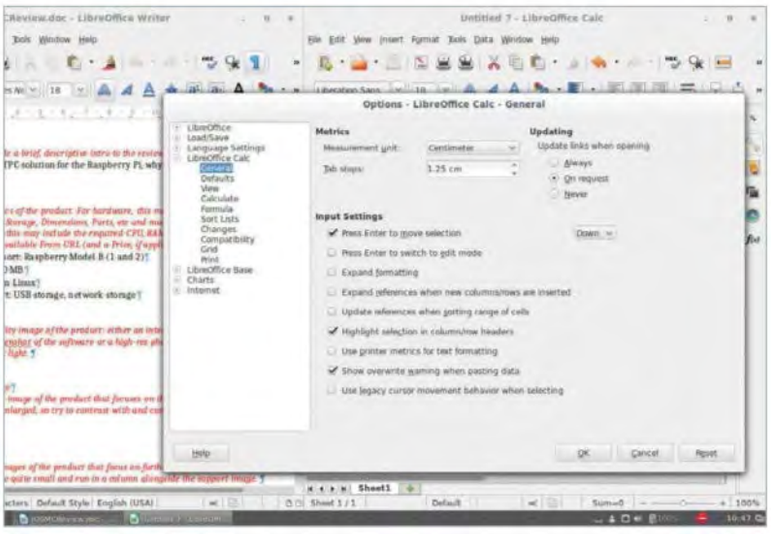
20 LibreOffice tips & tricks

Make your working day way more productive with these indispensable and powerful tips for your LibreOffice suite on Linux

Resources
LibreOffice
libreoffice.org

Using office software for very specific tasks, you can end up sticking yourself in a rut with the way you work. You do the same thing the same way every single time, not considering any other methods that might make it faster, better or more efficient. When you write several letters or documents a day for work, or maybe create spreadsheets with regularity, you don't really ever need to learn new techniques. We all get stuck in our ways.

It's sometimes difficult to comprehend just how powerful and feature-full LibreOffice can be, especially when you're treating it as you normally do and not making the most of what it has to offer. On a day-to-day basis, you may have no idea about macros, creating indexes or doing a mail merge with Writer. Over the next few pages, we are going to highlight some of the best ways you can improve your use of LibreOffice, with a particular focus on the core Writer and Calc programs. You'll optimise the way you work and start making the most of this feature-full office software.



Above Learn how to use the LibreOffice apps more productively

Writer

01 Bring up formatting

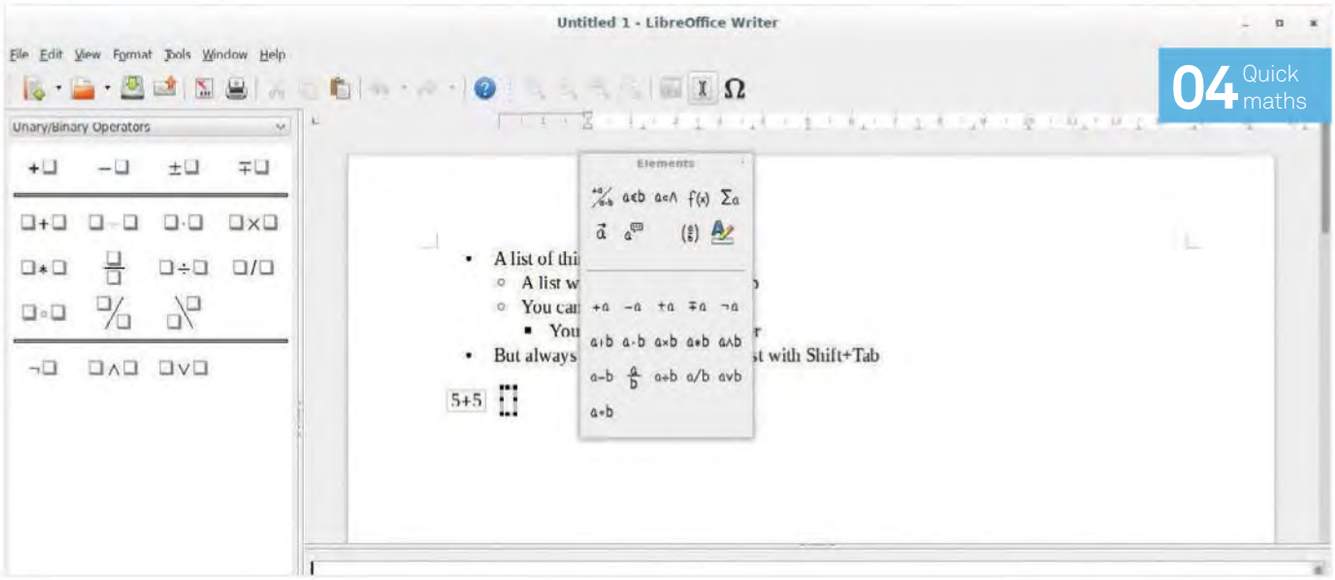
When formatting a piece, perhaps for greater readability or to make sure it prints properly, it can be a tricky to figure out exactly why certain sections of text are acting the way they do. On the toolbar is a symbol that looks like a backwards P – click that to reveal live formatting symbols, such as rogue paragraph breaks.

02 Better bullet points

Creating a list of bullet points is fairly easy; click the type with dots or numbers and go. You can change the formatting of these bullet points to be different symbols or letters instead of numbers. You can, however, also create nested lists by using the Tab key to create sub lists, and then press Shift+Tab to go back to the standard list.

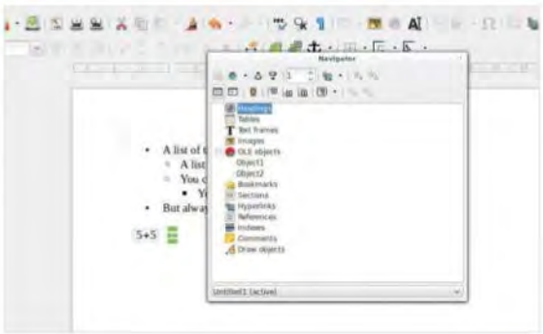
03 Paste unformatted text

Generally in Linux, you can use Ctrl+Shift+V to paste text and remove its formatting at the same time. If you use this in Writer, or find Paste Special in the Edit menu, you then have several options of how to paste the text. One of these is unformatted, but it also allows for other methods, such as using LibreOffice formatting.



04 Quick maths

Even when you're writing, you might need to do a quick bit of maths. Instead of switching to Google or a calculator app, you can use the formula bar. Go to Insert, Object, Formula and write out the calculation you want. Once you confirm it, the outcome of the formula appears where your cursor was placed.

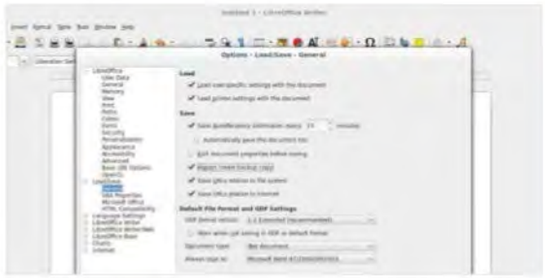


05 Quick navigation

Pressing F5 or going to the View menu will allow you to use the Navigator function. You can use it in large documents to quickly move between different headings, tables, graphics, bookmarks and many more objects in the document. It's not a proper dock, so you can move it around to see what you want at any given time.

06 Set default document format

By default, Writer will save new files as ODTs, the open document format. This works in most other word processors, but if you're regularly working with files that need to work on Microsoft Office, you can change the default file format to be .doc or .docx. Go to Tools, Options and find the General settings under Load/Save to change the default.



07 Create a backup system

Writer has a powerful recovery tool for if there is an unexpected shutdown of the software or computer itself, but that relies on temporary files and other related files that aren't always there when you really need them. Writer does have a backup system it can make use of though; enable 'Always create backup copy' in the Load/Save General options.

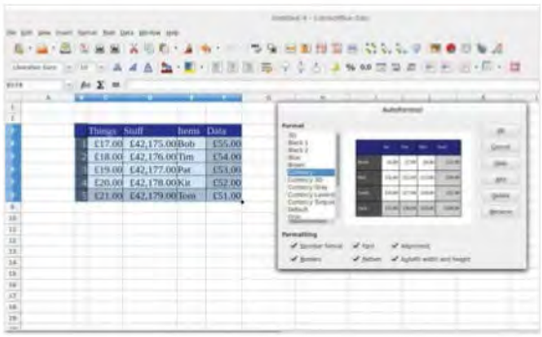
08 Change case

Written a sentence or a word and it's not capitalised correctly? Pasted some text and it's all randomly capitalised? You can actually change the case of specific sections of text without having to rewrite it – simply select what you want to change, right click and use the Change Case option.

Full app explanation

As luck would have it, if you're unsure of everything that comes in LibreOffice as part of the suite, we cover it extensively in our guide to the best Linux software starting on page 8. There is more to LibreOffice than just Writer and Calc, after all.

Calc

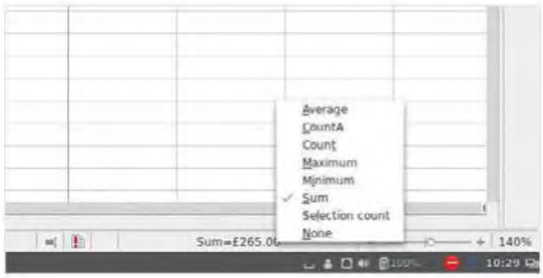


09 AutoFormat tables

If you've created a table, you may need to give it colours to make it more readable. You can do this manually, however Calc has a built-in format option under Format> AutoFormat. From here you can give a colour scheme to a table that you've selected and even customise what is taken into account for formatting.

10 Protect your spreadsheet

Shared spreadsheets are good for productivity, but the more complicated they become, the more difficult it can be to track down an accidental change made by one of the users. You can protect the spreadsheet by using Tools> Protect Document>Sheet to give it a password so that only certain people can make changes.



11 Change status bar preview

When you select some cells with a numerical value inside, you get a handy prompt on the status bar that tells you the value they add up to. That might not be what you want all the time, though. Click on that area of the status bar to bring up a menu and change it to average, maximum or minimum numbers in the selection.

12 AutoFilter rows

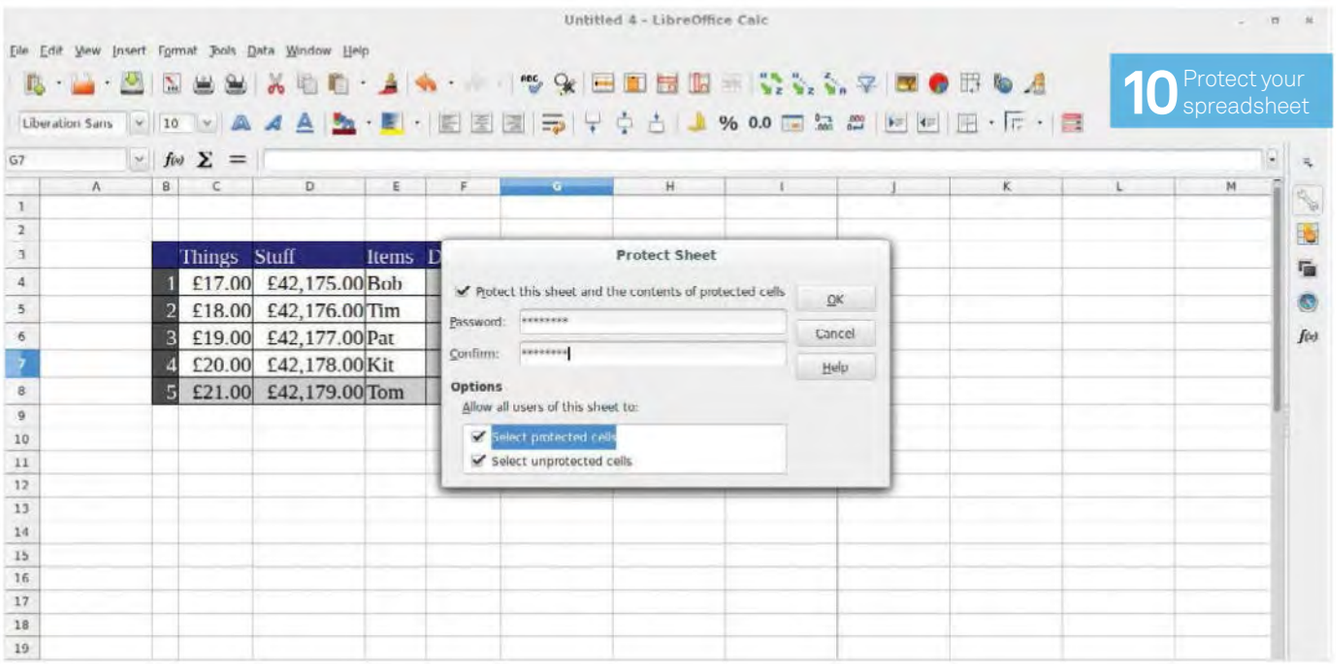
Filtering rows helps to organise data, but if you're not sure how exactly to go about doing that, Calc has an automatic filtering tool you can use. Select a row, then go to Data>Filter> Auto for it to create an automatic filtering system based on that row. You can also modify it a bit once it's in place.

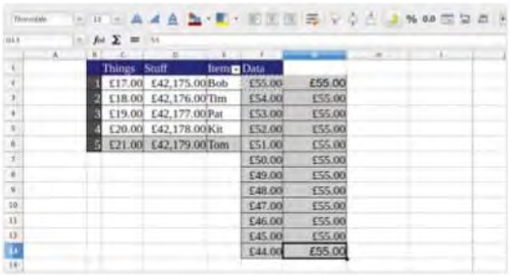
13 Grouped cells

Sometimes you don't need to see specific chunks of data all the time, and collapsing it like in a piece of code will add a bit more space to the viewable page. Using the Outline function under Data>Group and Outline>AutoOutline, you can create these collapsible groups, which use a thick outline to indicate themselves.

Learn more!

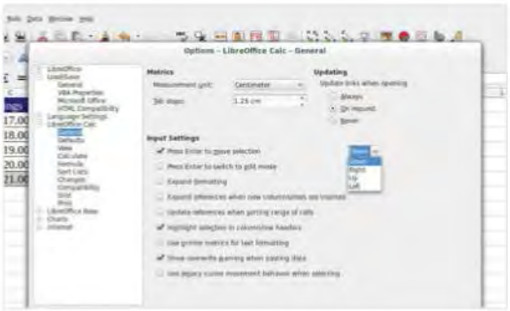
There are many more functions to learn that can help you do more with LibreOffice and the first step is to have a look through some of the documentation for the software, or just have a quick browse through all the options and the available menus.





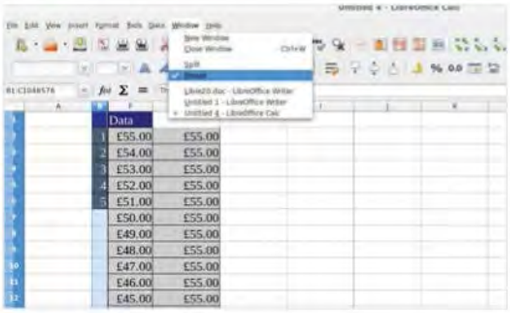
14 Auto-increment or copy cells

If you've never stumbled across it, Calc is very smart when it comes to replicating cells. If you write down two or three numbers or even dates, selecting one and dragging the black square down to copy will automatically fill in the cells with numbers or dates in the correct sequence. If you don't want it to do that, hold Control when copying the cell for regular duplication.



15 Change Enter key

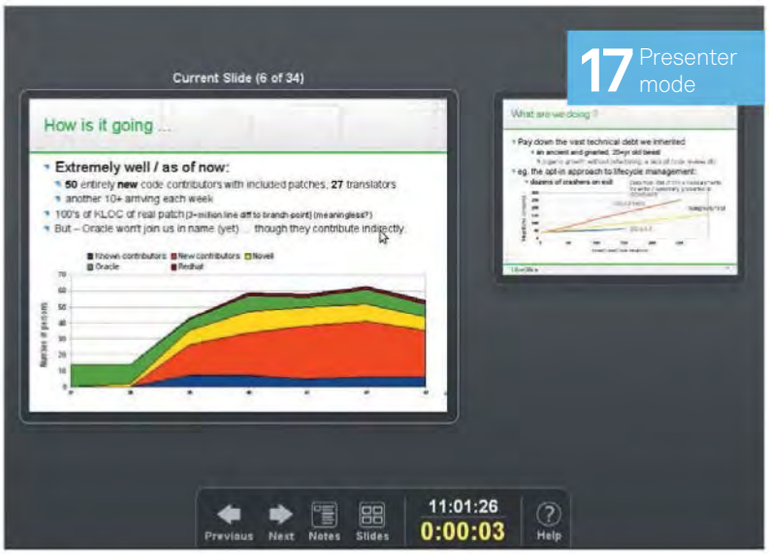
It seems fairly natural for the Enter key to move you down a row of cells and that's what Calc does by default. However, you can actually change the behaviour of the Enter key by going to Tools>Options>LibreOffice Calc, then General to have it move along a row instead.



16 Freeze columns in place

If you have a lot of data, rows or columns you can sometimes find yourself browsing the spreadsheet not always able to remember or divine what cell is for what. By selecting a row or column you always want to be visible, go to Window then Freeze to keep it always on the top or on the left as you browse the spreadsheet.

Miscellaneous



17 Presenter mode

When using Impress for presenting slides, you will often be hooked up to a projector or television that either mirrors or acts as an extension to your laptop. Impress has a neat feature where the actual presentation will be shown on the big screen, while you can turn on a presenter console just for your laptop display that shows you what's coming up in your presentation, along with notes.

18 Switch language spell-check on the fly

We'll illustrate this with an example: perhaps you need to paste into a document a paragraph from Napoleon that is written in French. The rest of the document is in English, so the spell check flips out at words from over the Channel. Highlight the paragraph, click Tools>Languages and then select a language for this section of the document, and only this section.

19 Insert readable formulas

There is a completely separate application for LibreOffice called Math that is not, in fact, database software, but actually a way to draw up mathematical equations that can then be inserted into other LibreOffice software. This is good if you're writing an academic paper with ridiculous maths that needs to be readable.

20 Using PDFs

PDFs can be edited in LibreOffice. Simply import them using something like Writer and it will dump the PDF into the Draw application. You can edit text, change pictures and even the general formatting of the PDF. Once it's done, Draw allows you to export the working file as a PDF for everyone to use.

Above Bodhi's biggest change is the new look, thanks to the Moksha desktop, but everything underneath works the same as before

DISTRO

Bodhi Linux 3.1.0

Jeff Hoogland's new Moksha desktop makes its first appearance as Bodhi abandons Enlightenment in favour of the new fork

CPU
500 MHz

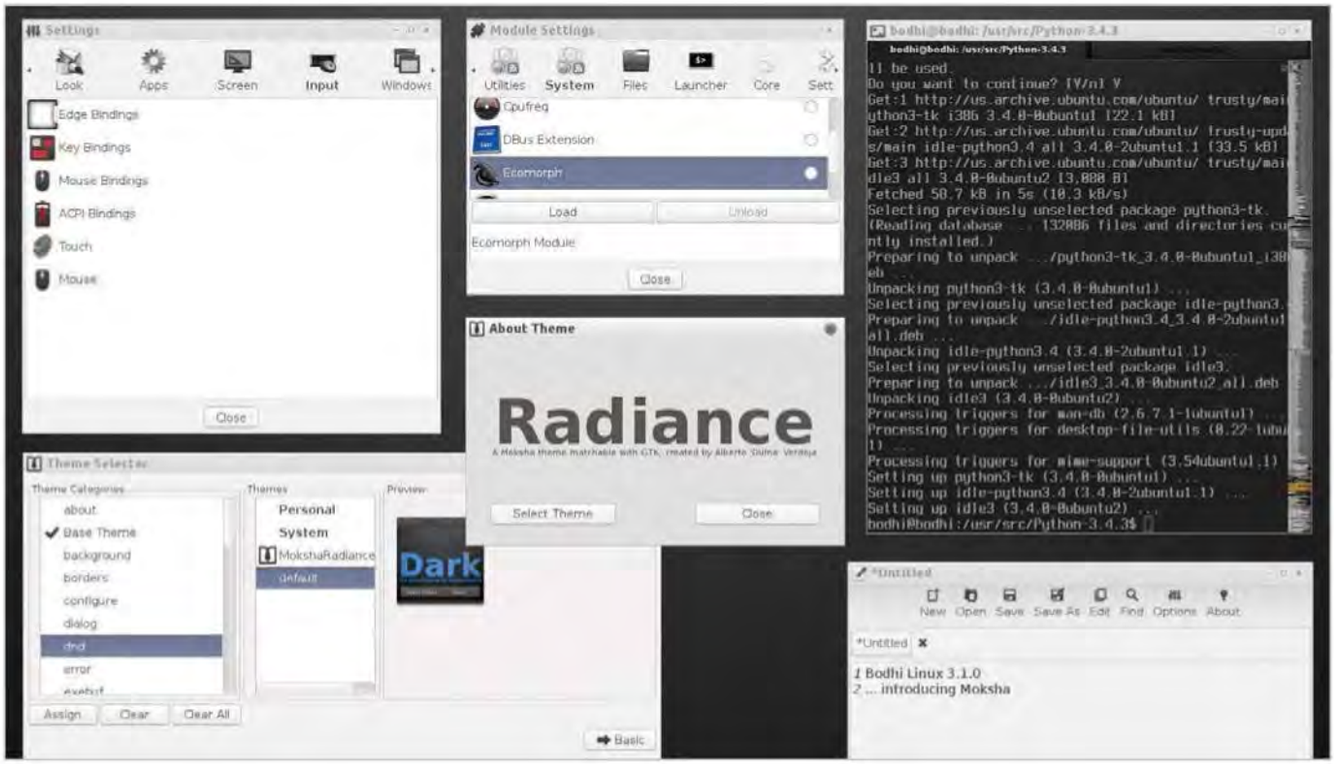
RAM
128 MB

Storage
4 GB

Bodhi Linux, based on Ubuntu LTS releases, is a highly workflow-based distro with two core tenets: minimalism and configurability. We talk about distros being minimal a lot, but Bodhi runs with this concept and provides absolutely sterling support for legacy hardware and low-spec machines. Additionally, the preloaded suite of applications is narrow, with Bodhi giving you just the ePad text editor, PCManFM, Terminology, Midori and ePhoto. Bodhi believes users are smart enough to choose their own applications – we're inclined to agree – and this means that the base apps total just 10MB of the whole image. There is a whole swathe of system tools, though, giving you an incredible amount of power to customise your system.

Much of the user choice is down to the desktop environment. With this new release, Jeff Hoogland, creator of Bodhi Linux, has introduced the brand new Moksha desktop, forked from the older E17 version of the Enlightenment desktop. Jeff was dissatisfied with the development work on Enlightenment, skipping E18 entirely because of a number of optimisations that broke crucial features from E17, while E19 was quite bloated and performed poorly on older hardware that was previously fine with Bodhi/Enlightenment. This led Jeff to release a Legacy version of Bodhi Linux 3.0.0 that stuck with E17 alongside the mainline E19 release. As a result, Jeff spoke to his users about their experiences and decided to fork the stable and preferred version E17.

As a continuation of E17, Moksha includes all of the patches that are usually made to Enlightenment for a Bodhi release. It's visually distinctive too, thanks to the new Radiance theme (matchable with GTK) that



Above The Moksha desktop is just as customisable as Enlightenment and retains the advanced theming options

> Moksha gives you absolute power over your desktop, so you can tinker with your setup to get it operating in precise ways

contrasts Enlightenment's iconic Dark theme with softer, lighter greys. Going forward, certain features will be backported from E18 and E19 – the aim is to achieve a better, more stable version of Enlightenment. Many of the features now removed from Enlightenment are present in Moksha, such as the advanced theme options that enable you to mix and match components from different themes, as well as the choice of modules that can be enabled for further functionality. Importantly, the memory usage is also significantly lower than with E19 and, as Moksha improves over the coming releases, we can expect it to become a refined, much lighter version of Enlightenment.

Booting Bodhi 3.1.0, the first thing you are greeted with is a Midori-launched QuickStart document that contains everything you need to know to dive right in. The Moksha basics page walks you through the concepts unique to Moksha/Enlightenment, such as shelves, modules, gadgets, the launcher menu that can be accessed from any point on the screen, and it also lists some of the essential shortcuts you'll need to learn. The shortcuts are often different from those that are standard to other desktops, potentially slowing you down if you're new to Bodhi, but commit them to muscle memory and you'll find Moksha really useful. There's also a great guide to Ecomorph in the QuickStart, the module that adds many popular compiz effects to Moksha. This sort of thing is where Bodhi's configurability really begins to shine.

Using the myriad settings panels throughout Bodhi Linux, you can fine-tune absolutely everything on your desktop: manipulate the window geometry, scaling, border and transition effects, plus their process management; you can load modules to add gadgets to your shelves, then place and reorder them as desired; you can change the key bindings, plus the mouse, screen edge and ACPI bindings.

Moksha gives you absolute power over your desktop, so you can tinker with your setup to get it operating in precise ways. It can be a little overwhelming for some new users, especially with various options and settings scattered across a range of different panels, but once you're orientated, you're on to a winner.

Pros

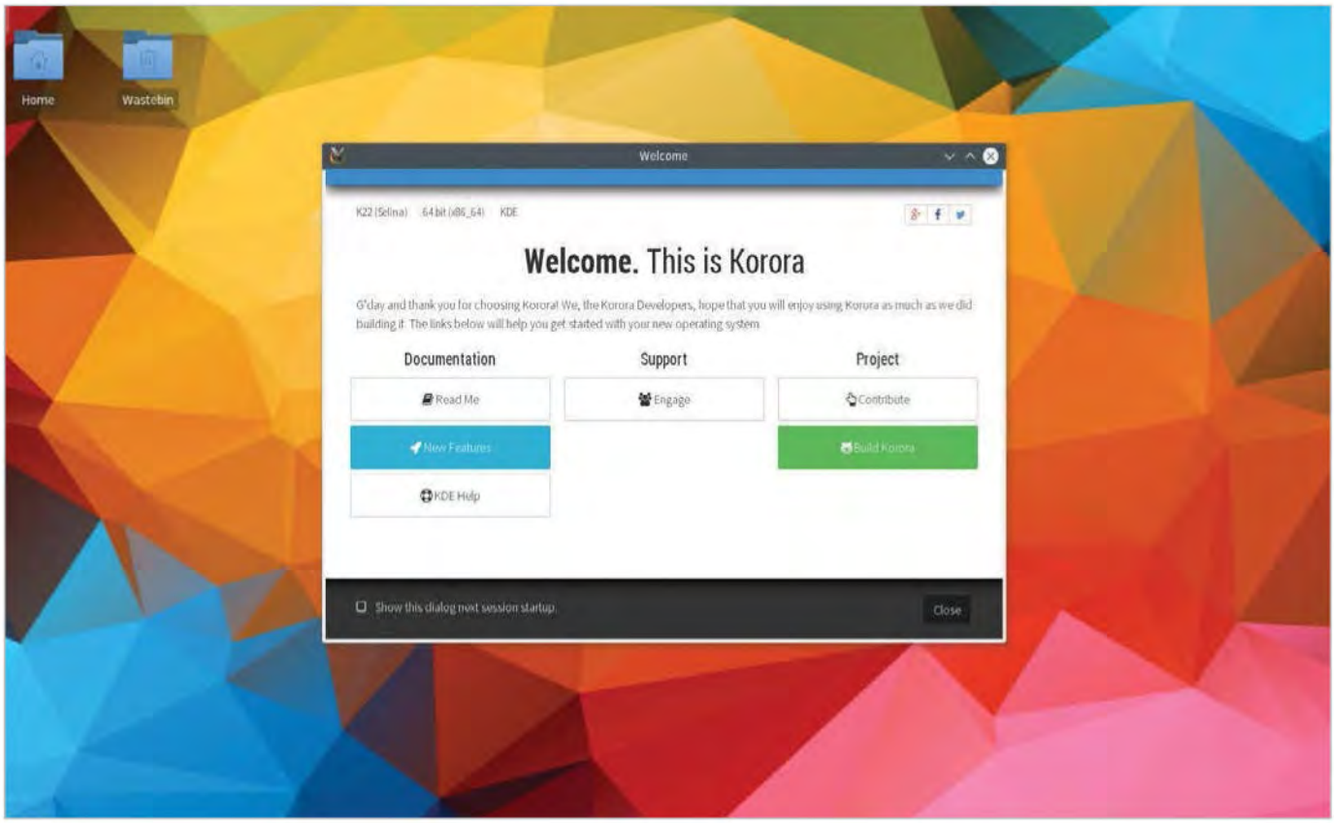
Enlightenment is excellent for customisation and Moksha preserves this functionality while offering a fresh take on E17

Cons

New users still need to invest a lot of time in the initial setup to get the best UX. The benefits of forking Enlightenment are yet to be proven.

Summary

Bodhi fans will like the well-crafted Moksha desktop and the preservation of existing workflows. They'll also benefit from added stability now that the desktop has forked away from the Enlightenment dev cycle. New users will discover a powerful and highly configurable system.



Above Korora 22 has an outstanding UX, from the aesthetics of the desktop environments down to the back-end settings and repo provision

DISTRO

Korora 22

Enhancing Fedora with popular software and sensible tweaks, Korora aims to "just work" right out of the box

CPU
x86

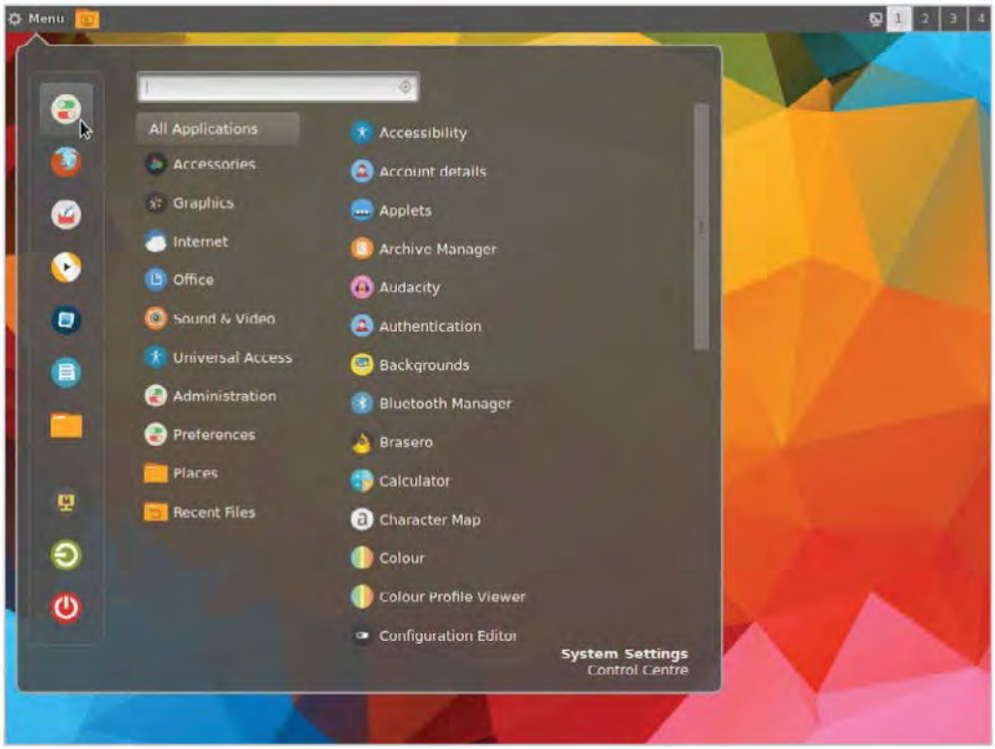
RAM
1 GB

Storage
10 GB

Fedora is a great distro, treading the line between stability and bleeding-edge updates incredibly well, but for the majority of people there is always a fair amount of post-installation prep to do before you can get down to business. Fedora ships out with free software only, and while this is fantastic for many Linux users and developers, there are also many who instantly look for proprietary software like Chrome, Flash, multimedia codecs and the like.

Korora takes the excellent Fedora base and then delivers what it believes to be the best and most popular customisations, from additional packages and repos through to system settings. Korora includes the RPMFusion repos by default, both the free and non-free; it comes with a base kickstart file that opens the firewall for Samba, SSH, Multicast DNS and printing, and also enables and disables various other services to better serve the home user.

The kickstart also includes some commonly used packages like Chrome so they can be installed out of the box. There are also kickstarts for the core desktop environments – KDE, Cinnamon, MATE, GNOME and Xfce – to provide desktop-specific software. One of the key Korora changes is that the developers tweak the RPM packages to make them persistent and so preserve the Korora customisations when these packages receive upstream updates. As a Fedora Remix, there are RPM replacements for Fedora-



Above Each available desktop comes with its own unique settings that help integrate it with the overall feel of Korora

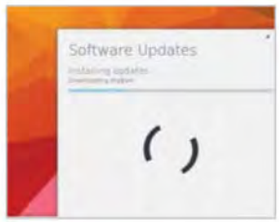
> Korora takes the excellent existing Fedora base and then delivers what it believes to be the best and most popular customisations

branded software, but as well as this the devs also provide the "release" RPMs that can be installed on the first boot without having to track them down online first. And they also add packages like Steam that aren't available in the Fedora or RPMFusion repos. In addition, there's a bundle of plug-ins for Firefox, such as Adblock Plus and DownThemAll, which is set as the default browser. VLC is your default media player, and while the distro uses free software where possible to help play your media, it doesn't shy away from the non-free multimedia codecs that many people find necessary. Korora also uses the Pharlap tool to help with the installation of third-party drivers such as NVIDIA's. The Korora devs really do an excellent job of considering the needs of their users.

With Korora 22, the major changes are predominately those inherited from the Fedora 22 base: Yum has been replaced with DNF and Hawkey, the Elastisearch indexing server is now in play, GCC 5.1 is the main compiler, and then there's the inclusion of Vagrant plus the upgrades to Ruby 2.2 and Perl 5.20. One notable change that wasn't taken from Fedora is the Korora devs' decision to no longer support Flash out of the box, joining a growing number of distros that have washed their hands of the flaw-ridden software.

The well-integrated desktop environments have also been inherited from the Fedora 22 release: we have KDE Plasma 5 with the excellent new Breeze theme, there's Cinnamon 2.6, GNOME 3.16, but the distro supports many more desktops like MATE, Xfce, Openbox, LXDE, Enlightenment and Sugar. You make your initial desktop choice when you go to download the ISO, but Korora makes it incredibly simple to switch desktops once you're up and running through the login screen's session manager.

You can use the Yumex GUI to browse through the desktop groups and install them along with the Korora-specific settings for each. You can also go via the command line by using **dnf install @yourchoice-desktop** and then **dnf install korora*yourchoice** to bring in the tweaks. Those per-desktop settings are excellent, too – Korora adds a GNOME Shell extension to GNOME that shows an application dock by default and it also adds the 'Open in Terminal' contextual menu option, for example. The cohesion of this distro really is quite something.



Pros

Fedora without post-install prep; useful system tweaks are preset, there's a huge range of packages and the desktops are customised

Cons

Inclusion of non-free software will divide Fedora fans. The Korora system and desktop tweaks won't necessarily appeal to everybody

Summary

Korora takes one of the most popular and stable distros and delivers a solid improvement, augmenting and adding to the Fedora experience without compromises (beyond the inclusion of non-free software) that could drive away expert users and developers. Korora 22 is one of the most polished, inviting and usable systems that we've seen in a long time.

Above The My OSMC menu can be customised to your liking

DISTRO

OSMC

It's yet another new HTPC solution for the Raspberry Pi, so why is OSMC worth a second look?

Hardware support
Raspberry Model B (1 and 2)

Image size
330 MB

Base OS
Debian Linux

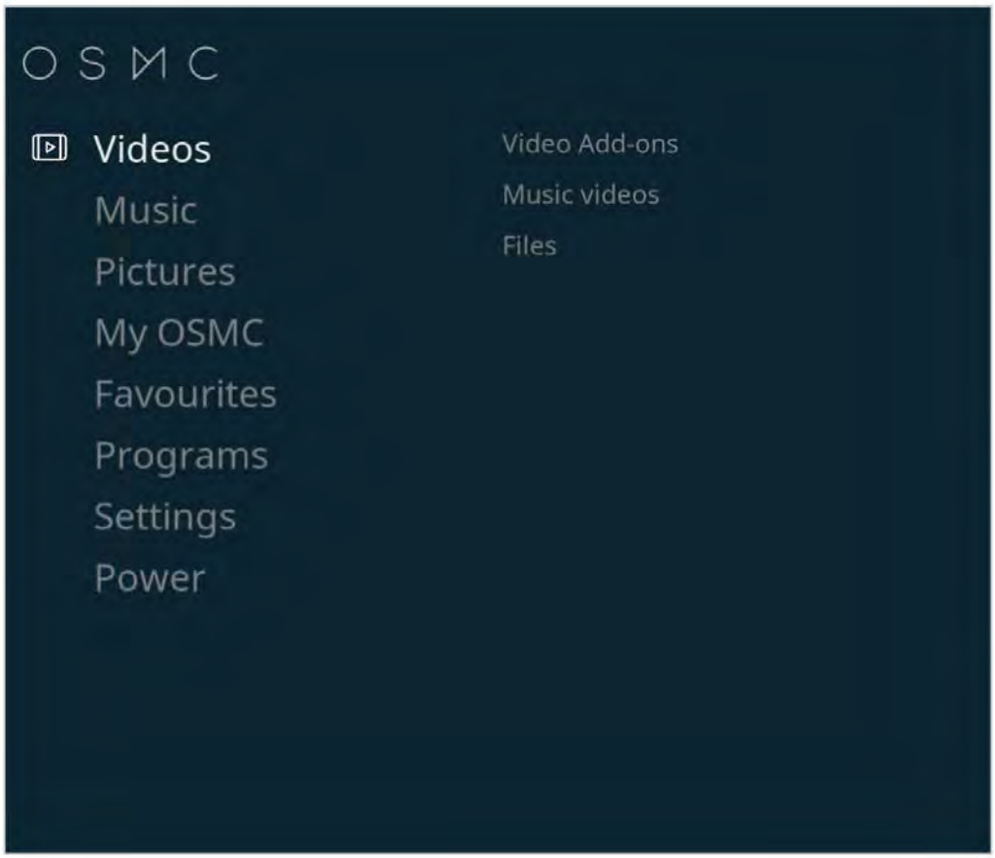
Storage support
USB storage, network storage

OSMC is finally 'complete', in a sense, with a stable release to its name. For something we've only covered once before in a small section of the magazine, it may not seem like such a big deal. It's a new piece of media centre software for the Raspberry Pi and we're well aware that there are plenty of those. It's also running Kodi, the new name for XBMC, which all the other operating systems use as well. In fact, you could just install Kodi yourself on Raspbian and have a functioning HTPC.

What sets OSMC apart though is the slight differences throughout. It's the successor to the original Raspberry Pi media centre OS, Raspbmc, from which the team has taken Kodi and given it a massive visual overhaul to fit their own vision. It's all on top of a Debian/Raspbian base, so this gives you not only a good, stable media centre, but also the ability to expand and customise the behaviour and functions of the Pi as it runs as your media centre, without actually affecting the running and operation of the media centre. This is something none of the other media centre solutions offer.

OSMC's strength lies in its interface. Although it's basically the same as Kodi in terms of flow and standard themes, the look and timing of the interface and browsing has been tweaked to feel smoother. The vertical interface showing all your options helps with ease and speed of navigation. The special My OSMC menu also contains the important settings in a custom interface that is slightly more logical to navigate than the standard Kodi one – this is also still in there, although it's a bit of a maze to get to the full menu. That said, it's the only really weird interface hiccup, though.

The initial setup for OSMC is nice and simple. It's available on NOOBS, and you can also dd it directly



Above The vertical interface enables quick and easy navigation

Pros

A great interface on top of the already excellent Kodi software makes this a good spin on the media centre

Cons

Settings menu and My OSMC are separated a little too much. This creates a weird split between some of the settings

> "Playback is good, playing just about anything you can throw at it with smooth 1080p decoding thanks to the hardware it's sitting on"

to an SD card. It's quite small so it doesn't take very long to write the card, making it worthwhile over NOOBS if you know how to do it. A lengthy, automatic installation phase takes place once the Raspberry Pi is turned on, followed by a straightforward profile setup. This includes Wi-Fi connection, something missing in standard Kodi and OpenELEC builds. It's an extra touch that may be minimal, but really adds to letting you quickly get into Kodi and start watching whatever you want.

The media adding system is the same as current Kodi. It will connect to local file sources, networked files through any number of services and also go online for information scraping services that will help you organise the files it can see. Playback is good, playing just about anything you can throw at it with smooth 1080p decoding thanks to the hardware it's sitting on. However, you can't navigate the menus as you are watching videos.

OSMC is good, then. Very good, living up to the potential we felt it had when we last looked at it. The tweaks to the interface make a world of difference, especially for something that you're using on a TV screen. The Kodi team seems to prefer to focus on the actual running of the software itself, which is fine, so to see a fresh take on the UX to augment the software itself is very welcome.

But should you be using it as your main HTPC OS? While it is definitely excellent, it's not a revolution. If you're building a system from scratch, we can absolutely recommend this, especially with a Raspberry Pi 2.

OpenELEC may be smaller and lighter, but OSMC is light enough, and the Debian core means you can have it pull double-duty and act as a file server as well as your media centre if you really want it to. It's most definitely worth a look before you settle on anything else, that's for sure.

Summary

An excellent media centre operating system that not only has a much better interface than anything you'd find on the other Raspberry Pi releases from Kodi, but also has a fully functional Debian base in case you want to do some further tinkering and customisation of your own. It's definitely one of the best Pi HTPC systems around and you should try it out for yourself.

Above Plasma 5 is available but not fully supported, so Mageia 5 defaults to version 4 of the desktop environment

DISTRO

Mageia 5

The company itself may be confined to history, but Mandriva Linux lives on through its true successor, Mageia

CPU
x86

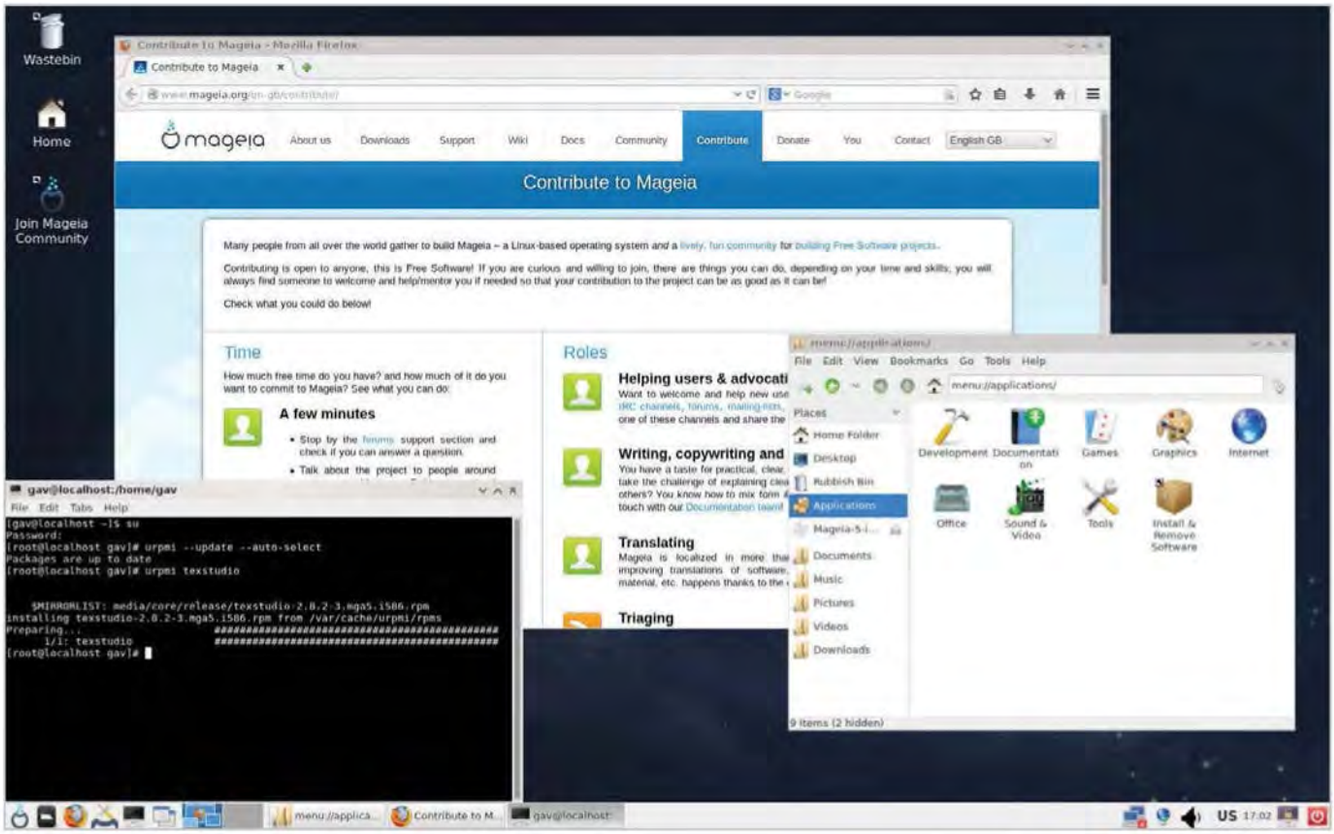
RAM
512 MB minimum,
2 GB recommended

Storage
5 GB minimum
20 GB recommended

Mageia 5 has been over a year in the making and, coming hot on the heels of the Mandriva liquidation, it definitely feels as though it is a timely release. The distro still bears the mark of its heritage in features like the Mageia (formerly Mandrake/Mandriva) Control Centre and the DrakX installer, with the former in particular being cutting-edge for its day. For this release, which developer Rémi Verschelde says may be its best one so far, both the Control Centre and DrakX have received significant updates, including both new features and fixes for bugs that date back to the old Mandrake/Mandriva days.

It's clear to see that an impressive amount of work has been performed on the installer, with a key new feature being UEFI support for modern machines. This little gem means that you won't need to use the workaround that you used for Mageia 4 if you faced this issue last year. This is an important update for Mageia as it makes the distro far more accessible to those with computers bought in the last three years, and it was mainly the efforts towards UEFI support that revealed most of the bugs in Mageia that have been fixed for this release.

There's now RAID support built right inside the installer, an updated partition manager (now defaulting to GPT rather than LVM for large disks), and Btrfs support. Grub 2 works better too, enabling you to detect other operating systems and add them to the boot menu, as well as being better integrated with features like an install script and fail-safe entry now being provided. Users can take advantage of all this in addition to the usual DrakX options – where if you opt for the DVD ISO then you get full control during the installation and can choose to add multiple desktop



Above Packages can be installed using Mageia's custom RPMdrake GUI or its urpmi command line utility

> An impressive amount of work has been done on the installer, with a key new feature being UEFI support for modern machines

environments, set your network boot options, change your shell or bootloader, customise the services that are activated by default, enable proxies and firewalls, set up your repos and very finely control which packages are installed by default.

The hugely comprehensive DrakX installer is one of the real draws of Mageia, though you can always opt for the more straightforward Live CD or instead the dual-architecture ISO for even faster, more expert setups (the latter is aimed more at sysadmins).

Once you're actually inside Mageia, we're pleased to report that it's really simple to get started. The welcome splash is a great orientation for new users, explaining how to add the non-free repos in case you didn't sort those out via your installation medium of choice, identifying and linking to the RPMdrake GUI and URPMI CLI tools for package management, and also providing a handy Applications section. Essentially a shortlist from the main package manager, this Applications section provides new users with quick links to audio and video codecs, plus popular software like Skype, Firefox, Flash, Smplayer and the like. With all these essentials covered right off the bat, you instantly feel at home in Mageia.

Some of the desktop environments are better implemented than others – we had a little trouble with Cinnamon while testing – but the ability to start off with a mix of KDE, GNOME, LXDE, LXQt, Xfce, MATE, Openbox, Cinnamon and Enlightenment installed, and easily switch between them at the user login screen, is excellent for quickly customising your experience. It's hard to fault the distro, really – the only thing we felt was missing was slightly more up-to-date software (GNOME is on 3.14 and without the new notifications system, for example, Firefox is the 31.7 ESR rather than version 38, and we have KDE 4.14.3 rather than the newer Plasma 5).

But as Rémi said, it had to stop at some point in order to get the distro stable enough for release – and it has done an excellent job.

Pros

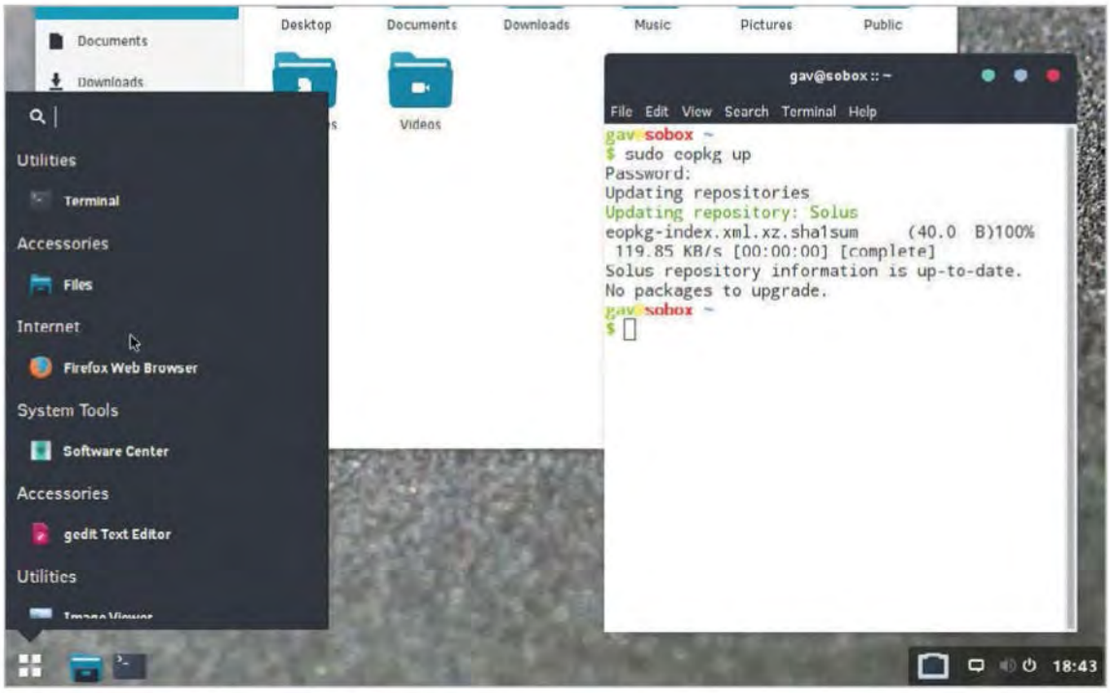
Crucially, UEFI is now supported, and the Mageia Control Centre and customisable DrakX installer have received significant upgrades

Cons

Huge range of package updates, but Mageia's longer freeze time has meant that this release is lagging in terms of version numbers

Summary

Mageia is an excellent distro and this release enhances its core system with under-the-hood improvements and fixes. Weak dependencies have been introduced to package management, while UEFI support makes the distro more accessible.



Left Clean, colourful and contemporary design are mainstays of Solus Operating System

DISTRO

Solus Beta 2

A fresh distro built from scratch with a brand new desktop environment called Budgie, Solus is designed to 'just work'

CPU
64-bit

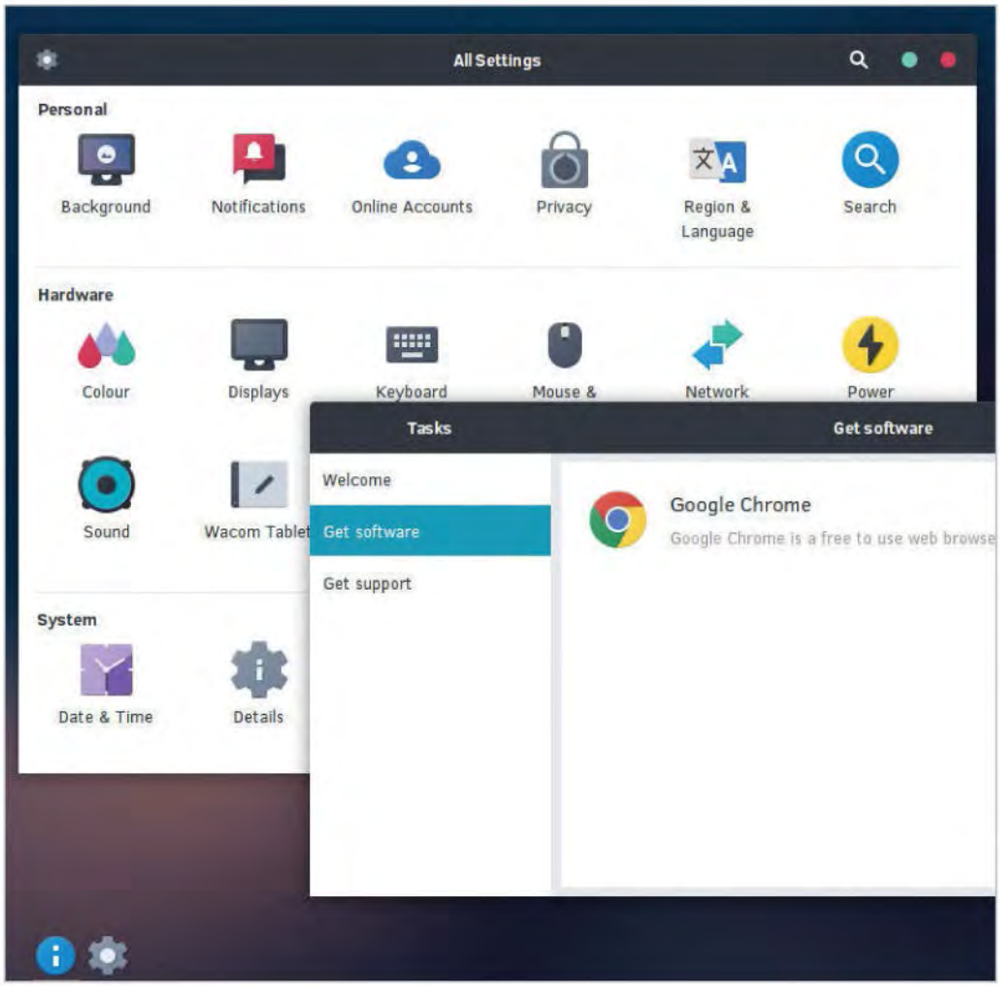
RAM
1 GB

Storage
4 GB

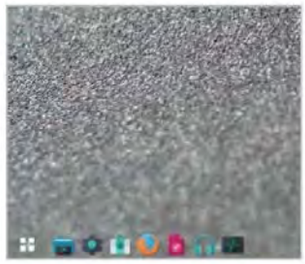
If the name sounds familiar, that's because the developer behind Solus Operating System, Ikey Doherty (of Linux Mint Debian Edition fame), released a distro of a similar name in 2012: SolusOS. Unfortunately, SolusOS disappeared in 2013, which was a shame because it came about in response to Ikey's work on LMDE, aiming to offer all the features that Ikey believed should have been in Mint's Debian Edition, and it was rather good. Ikey has, however, returned – the first beta of Solus arrived in 2015 under the name Evolve OS, which has been changed to Solus Operating System due to a trademark issue. While superficially similar, Solus and the original SolusOS are very different – while the latter was based on Debian, the new Solus is built completely from scratch.

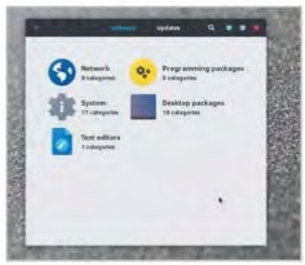
The first thing you'll notice is the custom-built desktop environment, Budgie. The desktop is tightly integrated with the GNOME stack, shipping out with the 3.16.3 version in this second beta release (on top of Linux 4.0.3). It is a refreshing, modern design, largely inspired by Chrome OS with some nods to GNOME and OS X, and it manages to offer a fresh new experience rather than purely imitate the aesthetics of other desktops. That said, Budgie also has a GNOME 2 theme if you want to use something more traditional-looking. While this beta is still working through some bugs, you can see that there is huge potential in this new desktop and it is undeniably gorgeous – it's available in some of the repos if you want to test it on your main computer. Budgie also gives you a decent amount of configuration and control without you needing to install any other tools, enabling you to switch icon and GTK+ widget themes, customise your panel applets and set up your menu how you like.

Things are less fleshed out on the software front, but that is understandable for a beta version of a distro in its infancy, which is also using its own package management system (eopkg, based on PiSi) and so needs time to grow its repos. Solus comes with some pre-installed essentials like Thunderbird, gedit, Firefox (though this is prone to crashing right now), a few system tools and Flash, plus some nice extras like



Above There's a welcome program to help get you started, though its software list is a little sparse right now





Pros

Beautifully made, a unique packaging system designed to reduce developers' workloads and great out-of-the-box configuration

Cons

Distinct lack of software and some system-wide kinks are yet to be worked out, although this is to be expected in a beta

> "Solus is specific to the desktop and 64-bit only, focusing on design, intuitiveness and software integration to evolve the concept of accessible distros"

extras like VLC, Rhythmbox and Transmission. Head across to the Software Center though and you can see that this is a work in progress – classics like Vim and Emacs are available, there's a growing range of system utilities, and there are also some programs like KeePass and even LibreOffice, though you'll need to manually search for the latter as it has not correctly been categorised as yet.

If you want to go beyond the initial lineup, however, you will have to wait for Solus to mature – specifically, for more developers to package their software in Solus' ypkg format. As a whole, the Solus Project is growing very quickly and Ikey is incredibly responsive to user requests and bug reports. UEFI and NVIDIA driver support was added for this release, for example, and there is a clear road map for a final version release in July. The new approach to package management that has been adopted also aims to make things much simpler for developers, with the ypkg tool simplifying down the packaging process itself by applying automatic rules and reducing the time it takes to complete.

We also like the general philosophy behind Solus – it is specific to the desktop and 64-bit only, focusing on design, intuitiveness and software integration to evolve the concept of accessible distros, creating a very reliable system for the home user that requires almost no configuration or explanation at all on the first boot. All that we can say is, elementary OS had better watch its back!

Summary

Developers and gamers will want to wait for the hardware and software support to improve. For light home use, though, Solus is ready to go now if you don't mind a few bugs – and even so, it's an excellent distro that we recommend trying yourself.


Unity

KDE Plasma 5 VS Gnome | LXDE | Unity | Cinnamon & more

We pit Plasma 5 against the other nine best desktop environments to see if KDE comes out on top, and which is the best fit for you

Roll back a few years and the Linux landscape looked very different. The last time we took a good look at desktop environments to see which was worth your screen space, it really was just a two-horse race between GNOME and KDE. Since then, GNOME has changed drastically, giving rise to a touch-oriented desktop that catalysed the emergence of MATE for those who longed for the good ol' days of GNOME 2, while KDE had a slightly less than smooth transition from Plasma 3.5 to Plasma 4 that left many nervous about Plasma 5. As a result of both of these evolutions and the emergence of some compelling alternatives, the arena is no longer dominated by these two heavyweight desktops alone.

But despite this, the recent launch of KDE Plasma 5 was one of the most exciting and talked-about desktop environment releases in a long while, putting the spotlight again on the GUI. With KDE being the mainstay of many a Linux user's machine, the real question now is whether or not Plasma 5 is the best desktop available today – and if not, why not, and what are the viable alternatives?

So, over the next few pages we're going to be running through the competition and showing you that in addition to the heavyweights, there are also some lightweights, newcomers and underdogs that are worthy of consideration the next time you decide to refresh your computer with a brand new desktop environment. To help you decide which one is right for you, we're going to be analysing each desktop in terms of its design, workflow, ease of use and its philosophy, as well as identifying which types of use each is most useful for. So if you're ready to renew your desktop, read on to see how our ten champions fare in this desktop battle royale!

KDE

The beautiful desktop has just received a major upgrade that reflects more modern aesthetics

Pros
Beautiful design, unique approach to app and file management

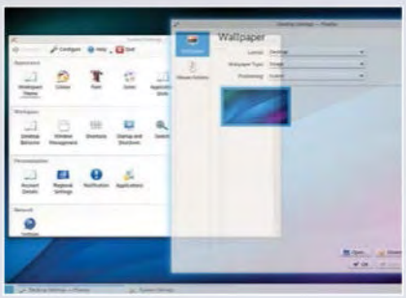
Cons
One of the most resource-heavy desktops with weak search capabilities

Best for
Home leisure

Associated distros
Kubuntu, openSUSE, Mageia

Workflow
Uses the traditional desktop metaphor, with a window bar and program menu. Files, folders and widgets can be organised on the desktop as well

http://kde.org/



To us, KDE has always been the desktop that has had the best aesthetics. Everything is uniform and beautiful. However, the designs sometimes get stuck in the time they were made in, and KDE Plasma 4 looks a little bit dated in 2014. However, that's all about to change as Plasma 5 has launched with a brand new look and a slightly different workflow.

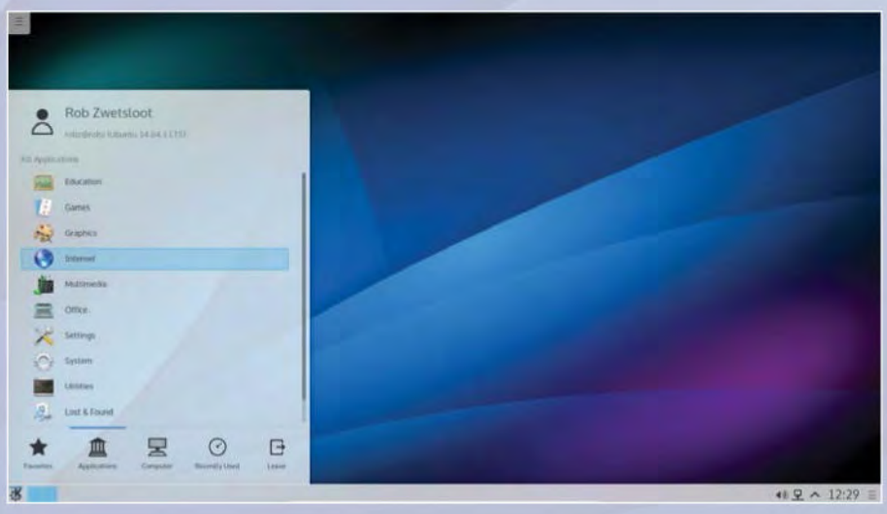
KDE versus GNOME was the desktop battle of the previous decade, with sides apparently leaning towards GNOME but with plenty of reasons why you should check out KDE. In the end, neither really won and, while still very popular, they're no longer in the zeitgeist like your Cinnamons or Unitys. Plasma 5 may not be quite the spark to get the old rivalry going again but it's definitely an interesting update to the desktop environment.

Firstly, the workflow is roughly the same: a mouse-led workflow for organising windows on the desktop and on the bottom panel, listed in the traditional method. KDE popularised widgets on the Linux desktop and they remain in the latest version of the desktop, including the interesting folder view from Plasma 4. While this is not activated by default like the previous KDE, it can be added like any other widget to better organise your files and desktop.

The new aesthetics for Plasma 5 are really very nice. Crisp, clean straight lines of modern computers and extremely well-labelled in the process. It's absolutely wonderful to use when it's working and reminds us of using Cinnamon for the first time when that was the king of design, and KDE 4 before that.

It is still a touch buggy, though, and doesn't play nicely with other installations of KDE 4 on the same distro. It's also not readily available through repositories yet, so you'll have to go looking on Google for instructions on how to install it – for major distros this usually involves downloading installation files, or adding a repository/PPA to your system and then installing it from there. Otherwise you need to look into compiling it from source.

Plasma 5 is definitely one to keep an eye on as it matures over the next six months, and we're interested to see if new usable features will be added rather than design overhauls.



Above The KDE design is beautiful, taking the clean lines of other modern desktops across all platforms

Plasma 4

Plasma 5 is still brand new and may take an iteration or two of your favourite distro for it to make it into the repos. It's also not 100 per cent stable at the moment so you may have better luck trying out the previous version of KDE: Plasma 4. It's still being developed and updated, and includes a very interesting way of organising open folders on the desktop for quick access and tidying. Right now, Plasma 4 is the version of KDE you'll find in most of your repos under KDE.

GNOME

The veteran desktop environment is forging a new path of its own that not everyone agrees with

Pros
A stable, mature codebase popular among developers

Cons
A quite different interface seemingly designed for touch devices

Best for
Development, touchscreens

Associated distros
Fedora, Red Hat Enterprise Linux

Workflow
The system is designed for both touchscreens and keyboard-heavy navigation via shortcuts, reducing the emphasis on the mouse

www.gnome.org

Above The search function in GNOME was one of the first to be implemented into a Linux desktop environment

The GNOME desktop is probably the most well-known desktop in all of Linux, in no small part due to its extreme popularity during the last decade. It was the desktop of choice for a lot of distros, although there wasn't as much choice as there is now, meaning most Linux users active in the Noughties will have encountered it at least once in their lifetime.

Things change, though, and GNOME received a radical overhaul in 2011 that didn't go down so well with several vocal members of the community. The all-important workflow of GNOME 2 was dropped for a brand new design that relies heavily on keyboard shortcuts along with mouse control. Over the intervening years, features have been added that imply a touch-focused interface reminiscent of the way you'd control running apps and windows on a smartphone or tablet.

The current philosophy behind GNOME seems to be a drive towards simplicity. Many native apps only contain the bare minimum features for the task they perform, such as the GNOME browser and the network managing tools. When maximised, windows lose the ability to be exited via the classic x symbol in the corner. In fact, normal windows do not have a maximising or minimising button. Toolbars are accessed from the top panel in an effort to keep things neat and tidy, to possibly increase screen real estate, and generally give the whole desktop a more smartphone-esque appearance.

Above Exposing the desktop with the windows button is a key part of a quite different workflow process

There are a number of issues that this can cause. As the design goes against the traditional 'desktop metaphor' seen in most desktop environments, it can take some time for people to adjust – especially if you sometimes need to go back to more traditional desktops in the meantime. The reliance on keyboard shortcuts also means that mousing around the desktop requires many more actions than before.

With a bit of practice and know-how, learning the new workflow and switching out the default apps can lead to an excellent desktop with advanced search capabilities. It will still have a few quirks about it, and there's no real way to use it on a touchscreen just yet, but it may well be a forward-thinking move if technology does go that way.

Common shortcuts

Windows button/super key
Opens the Activities overlay, which accesses the different apps and gives an overview of the desktop. You can immediately search for software and files or quickly close/move between windows

Alt+`
Switches between windows of the same application – this is handy if you have Alt+Tab switching between apps

Ctrl+Alt+Up/Down arrow
Switches between workspaces

Ctrl+Alt+T
Opens a terminal emulator

Best for
Family computer, media PC

Associated distros
Ubuntu, LXLE

Workflow
Use of search and the HUD to perform actions as well as regular browsing of the side bar to click between apps. Good for touchscreens

https://unity.ubuntu.com/

Unity shortcuts

Windows button/super key
Opens the search bar and lens as well as bring up the sidebar if you have it set to hide. You can type to search or click around to get what you need from there

Alt
Activates the HUD to search for commands with the toolbar of open applications or in-focus windows

Alt+`
Switches between windows of the same application, as Unity sets software switching by default for Alt+Tab

Ctrl+Alt+Up/Down/Right/Left arrow
Switches between workspaces which are organised in a square grid pattern once they've been activated

Ctrl+Alt+T
Opens a terminal emulator

Unity

Ubuntu's desktop environment is part of a distro and device-spanning concept

Pros
Desktop search hooks into online services and system tools

Cons
Advertising is built into the system and there are privacy concerns

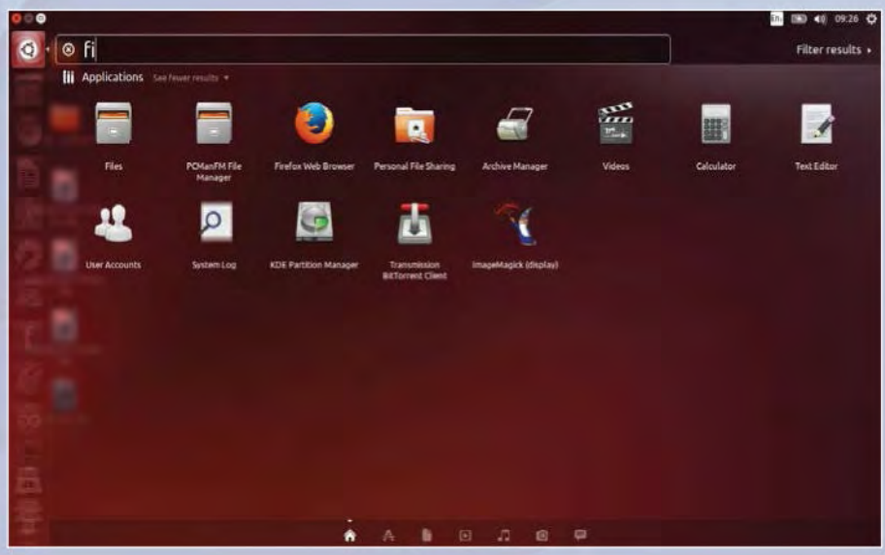
One of the most controversial desktops to date, Unity is part of Canonical's grand vision for an overarching Linux distro – in this case Ubuntu – across several types of devices. Being familiar with the desktop means you should be able to get started straight away on the phone interface as well, or at least that's the theory. Launched in 2010 and becoming Ubuntu's default desktop in 2011, Unity is a graphical shell that sits on top of GNOME and was originally implied as an alternative to GNOME 3 when it was released (to divisive reception around the same time).

Unity takes the search aspect of modern desktops and tries to split it up into categories for easier navigation, which is particularly handy if you're not sure exactly what you're looking for. What kind of music are you feeling in the mood to listen to today? Maybe you should scroll through the music tab to find out. Even if you do search, you can then easily drill down between these lenses and even add more as you see fit. The launcher on the side acts as a quick launch and a way to switch between open applications, with a focus on application-switching rather than window-switching by default.

There are also some extra minor features like the HUD, a way to access items in the toolbar of any open or in-focus applications instead of mousing through menus. It's an interesting system but, as a lot of workflow is based on muscle-memory or only remembering the vague description of menu items, it's not always helpful.

It's very much one of the more touch-friendly desktops, so much so that it almost makes the mouse seem like a hindrance to the operation of the desktop. Much like GNOME it can be keyboard-heavy in a lot of senses, however it generally feels a little more usable than the current GNOME shell because the mouse isn't completely obsolete.

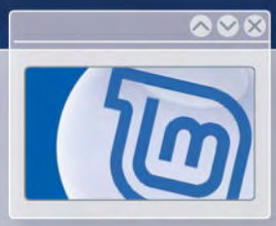
While Unity is heavily tied to Ubuntu it is open source and said source is available for you to use. It's not regularly packaged in repos, though, so you will have to find a third-party repository or build it from scratch.



Right The Unity search uses different scopes to make searching easier – these can be customised and more scopes can be added

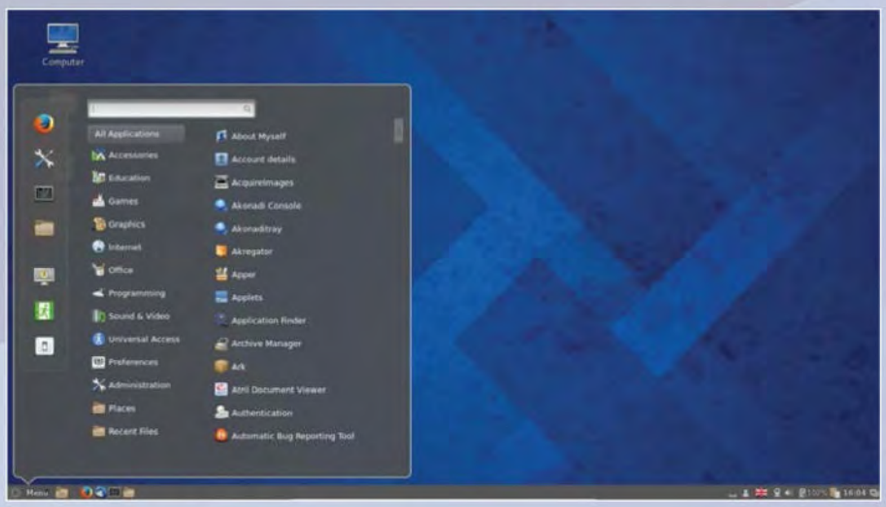
Cinnamon

The former GNOME fork that has emerged as one of the more powerful and flexible desktop environments



Pros
Straightforward and smart design using traditional desktop metaphor

Cons
Still not available everywhere and can also be quite resource heavy



Above Cinnamon started as the GNOME shell in a more traditional configuration and has evolved into a unique thing

Linux Mint has been around longer than the GNOME shell or Unity, and in fact used GNOME 2 as its default desktop for a time. Like Ubuntu, once GNOME updated the desktop to its current version, the team at Linux Mint decided they wanted to use something slightly different.

After trying out a release with the Mint Gnome Shell Extensions (MGSE) to bring the desktop to a more recognisable state, the team forked GNOME to create the Cinnamon desktop for the following release. The design ethos of Cinnamon has been to take the advanced features of GNOME – namely notification integration and desktop search – and package them in the more familiar format of the so-called traditional desktop metaphor. This includes an application menu and open windows listed on the panel at the bottom.

In addition to these basic features is a highly customisable applets section that lists and accesses specific services such as networking, updates, notifications, display resolution, and more. Features can be added and removed from here and in recent updates they will smartly sense when a different application is handling some of these core tasks and make sure icons don't double up. Cinnamon is very much about not compromising. There are no features included just for the sake of a philosophy or corporate deal. There is no unnecessary branding, and the interface is quite minimalist in size but not function. Everything is neatly labelled and presented to maximise finding your way around the desktop and file manager.

It's most certainly not touch-friendly, though. Menus are small and designed to maximise screen real estate, making buttons difficult to press if that's how you want to use them. In terms of keyboard shortcuts, the basic super key to open the search and Ctrl+Alt+direction key still work as they do in GNOME. However, Down and Up expose everything on the workspace or across all workspaces respectively.

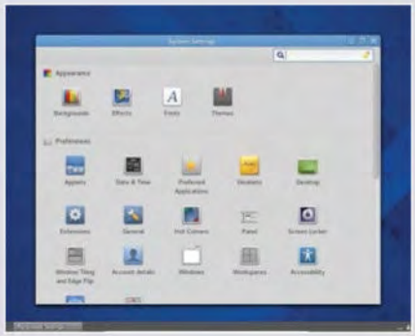
Cinnamon is now no longer a fork of GNOME, having become its own desktop and not just a shell around this time last year. This means it can grow beyond the boundaries set by GNOME and optimise the underlying code a bit more. However, it does result in a slightly less mature codebase. There are no major issues with it now, though.

Best for
Office, everyday computing

Associated distros
Linux Mint, Antergos

Workflow
Traditional desktop metaphor relying on mouse navigation with windows on a taskbar but with extra keyboard shortcuts to facilitate navigation

http://cinnamon.linuxmint.com



Desktop exposure

Using Ctrl+Alt+Up or Down simulates the effect of the Windows or super key in GNOME, bringing up an overview of the desktop. Pressing Up allows you to look across all the desktops and what windows are open in each – you can even name desktops from here or add more. Pressing Down or moving your mouse to the hot corner in the top-left will arrange all the open windows on the current desktop in a grid, allowing you to choose between them or quickly close them down.

XFCE

The first major lightweight desktop takes some design notes from GNOME 2 but forges its own path

Pros
An interface designed for the GNOME 2 generation

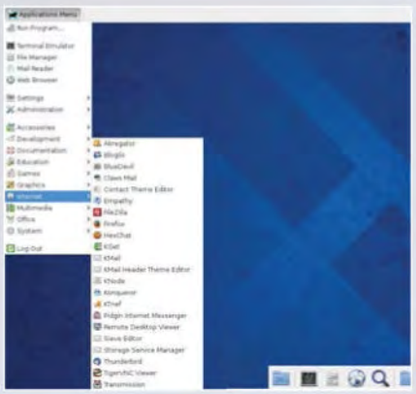
Cons
Limited by being lightweight, lacks any fancy search or desktop effects

Best for
Old computers, veteran users

Associated distros
Xubuntu, Debian

Workflow
A simplistic design utilising panels and a program menu that's best manipulated by mouse, with open windows listed separately

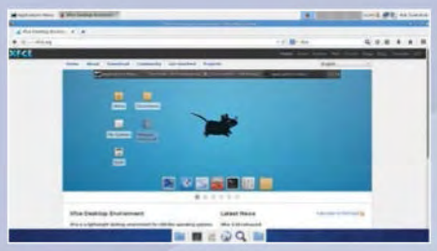
www.xfce.org



With its built-in GTK+ 2 toolkit like the previous version of GNOME, XFCE apes its panel layout and basic workflow. It's one of the oldest lightweight desktop environments around and more than likely popularised the concept once it was used for Xubuntu in the late Noughties. It's not the most well-known light desktop, with LXDE winning the popularity contest between the two these days. XFCE still has its loyal followers, though, due to being reminiscent of the older GNOME 2 design and being one the few desktops with a dock bar.

XFCE is comprised of said dock bar as well as a main panel on the top of the screen. This includes a program menu that not only includes the main software categories but also contains separate categories for system settings and preferences. Open windows are listed on this top panel while quick links to a customisable selection of apps is located on the dock bar. The dock bar doesn't hide by default; it acts like a border for open windows, reducing screen real estate unless it's removed or the hiding feature activated. You can also make the top panel hide to truly maximise screen real estate.

As for lightness, recent benchmarks put XFCE at about 89 MB of memory, which is slightly higher than LXDE. XFCE is slowly making the update to GTK+ 3, which will increase this number. For now, though, it's still a lot lighter than the likes of KDE and Cinnamon and will likely still be significantly less resource-intensive even when on GTK+ 3. It's also slightly more functional than LXDE, with more going on in terms of the panel and the dock bar and more. Choosing between the two is very much a balancing act and depends on whether or not your system can spare a few more megabytes for a slightly more usable desktop.



Above The dock bar can cause screen real estate issues if you don't change it to autohide

Above XFCE will look familiar to GNOME 2 users. You can switch out the dock bar for a panel to simulate it better

Dock options

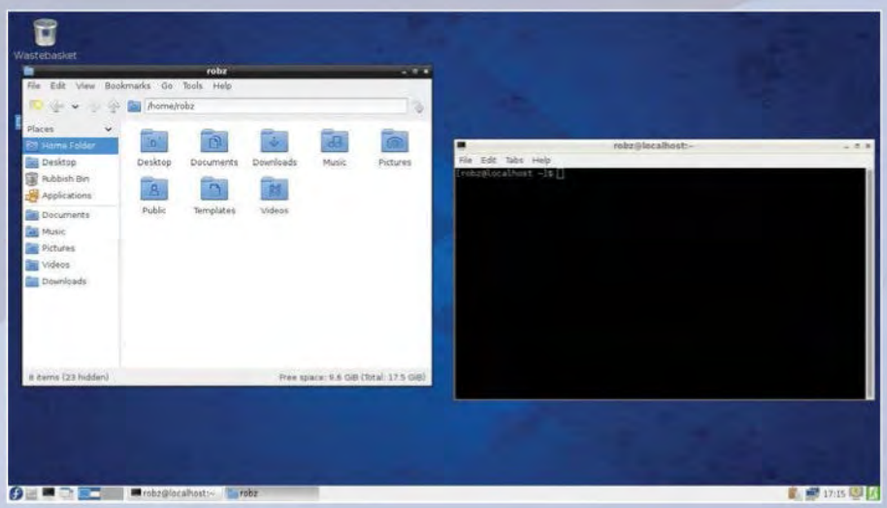
The dock may not be to everyone's liking – either you find it unnecessary or you want to make it hide when not in use. To do this, you need to find the panel settings menu in the XFCE settings; open this and select Panel 2 from the drop-down list. From here you can select the option to 'Automatically hide and show panel' which will remove it when it's not in use, or remove the panel entirely by clicking on the minus symbol when it's selected.

LXDE

The lightest desktop that doesn't sacrifice usability for the sake of a few extra cycles

Pros
The lightest distro with the most recognisable interface

Cons
Some basic, albeit non-essential features are removed



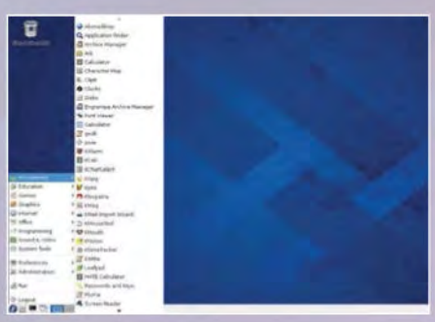
Above Window managers can be changed to eke out every last megabyte, but the default one is pretty good itself

LXDE is so ubiquitous with being lightweight it's in the name. The Lightweight X11 Desktop Environment is newer software than XFCE, debuting in the late Noughties when GNOME and KDE were still the top two desktops around. Like XFCE it's built on GTK+, but it resembles KDE 3 more with its take on the desktop. It's about as minimalist you can get without actively losing the features of a standard or traditional desktop. It's so concerned with resource use that a CPU monitor is incorporated into the panel by default.

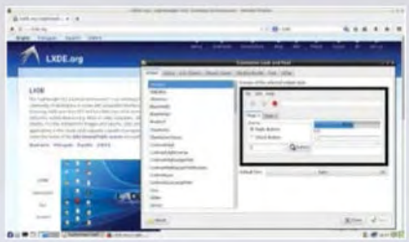
LXDE is a very mouse-driven desktop. There are no important keyboard shortcuts and everything needs to be accessed via point-and-click. Windows are typically arranged on panel and the window styling is incredibly minimalist. The program menu holds the standard range of categories for programs and access to settings and such. All the default LXDE apps are quite basic but they're all quite customisable in the process. There's nothing really quantifiably special about LXDE in terms of workflow – it's just that it has a good basic workflow while being as lightweight as it is.

Specifically, LXDE uses only 78MB of RAM, which is 20MB on top of Openbox as a pure window manager. While this is only 11MB less than XFCE, if you're only running on 256MB of RAM, that's a big enough amount to care. This is why LXDE is used on Raspbian, as the Raspberry Pi really cannot handle much.

In terms of eking out every last drop of a more powerful system, the gains made between XFCE and LXDE are extremely negligible as you're getting into serious diminishing returns when you're concerning yourself over ten megabytes on an eight gigabyte system.



Left Another simplistic program menu that does no more than it needs to



Best for
Very old systems, underpowered and single-board computers

Associated distros
Lubuntu, Raspbian, Knoppix

Workflow
LXDE uses a minimal traditional desktop metaphor, with a panel at the bottom with access to programs and a mouse-orientated navigation system

www.lxde.org

New file manager

LXDE can be further modified to make it even lighter than its standard version, namely by changing the file manager. While Openbox is plenty light itself, there are other file managers that do just about enough and save on a few extra megabytes. Here are some suggestions on alternate file managers:

IceWM
A lightweight manager sometimes paired with LXDE on extremely lightweight systems

FVWM
A much older windows manager that was made in a time when resources were much more precious

Enlightenment
We talk about this later as a full-blown desktop but it can also be used as a lightweight windows manager in LXDE itself

MATE

A fork and continuation of the GNOME 2 code

Best for
Veteran users looking for modern features

Associated distros
Linux Mint, Snowlinux

http://mate-desktop.org

Pros
A lightweight continuation of the popular GNOME code

Cons
A little stuck in the past for some users and can't compete with LXDE or XFCE

Cinnamon's lighter counterpart is the continuation of the older GNOME 2 line, forking the code when some users were unsatisfied with the direction of the GNOME Shell. The MATE team's design goals are to keep the GNOME 2 style of desktop alive and up-to-date, incorporating modern desktop features such as notifications and application/file search.

It's best for veteran users that prefer the workflow style of the classic GNOME 2 desktop: everything accessible from the top bar without hunting through separate apps, windows displayed on the bottom panel and a mouse-centric design. It's currently in a fair few distros and is being added to more all the time. It's much lighter than the likes of GNOME, KDE and Cinnamon too, but can't really compete with LXDE or XFCE.

Above By default you can make MATE look exactly like the old GNOME 2, although Linux Mint uses a slightly different configuration

Enlightenment

An alternative lightweight option with a unique yet familiar look

Best for
Highly custom, light setups

Associated distros
Bodhi Linux, Elive

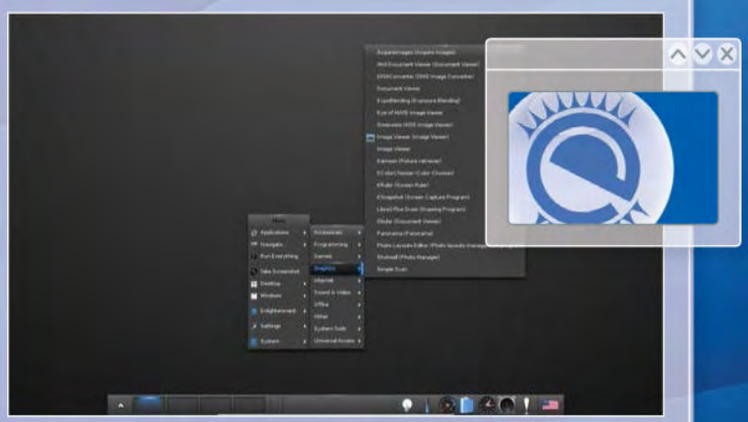
www.enlightenment.org

Pros
Lightweight and unique with a focus on usability and options access

Cons
Unique interface may be confusing to utilise for some of the users

Enlightenment relies on a slightly atypical panel on the screen that contains system info and a list of open windows. The program menu is accessible from this panel, but right-clicking anywhere on an empty part of the desktop will also bring up the options. As the panel doesn't properly take up the whole bottom of the screen it actually seems like you're missing out on screen real estate, but Enlightenment uses it quite well.

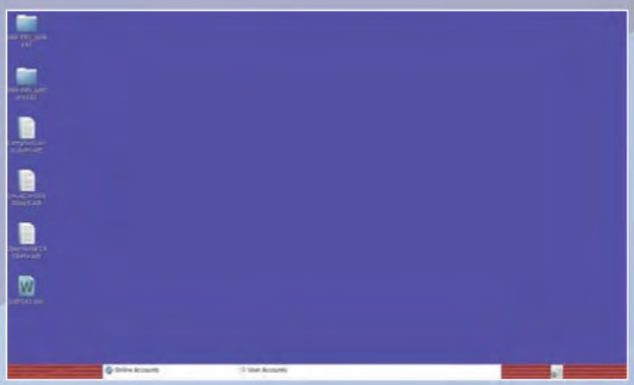
It's very fast, light and mouse-heavy in its use. Due to the way the program menu is hidden away, it's not the best candidate for a touchscreen, and it doesn't really have a range of useful keyboard shortcuts either. It's a very overlooked desktop that doesn't use GTK or Qt to power it, making it much better at running a mixture of differently coded apps.



Above You can access the application menu from anywhere you want, just in case you can't quite get down to the panel

LXQt

The Qt port of the lightweight desktop has some visual differences



Above The aesthetics of LXQt are closer to KDE than anything else, but despite that it's still LXDE

Pros
Lightweight and the same old LXDE with a fresh coat of paint

Cons
LXQt is actually a little heavier on resources than LXDE

A port of the excellent LXDE to Qt instead of the GTK it's currently built on, LXQt is also the successor to a desktop called RazorQt that we had the pleasure of using for a period last year. It certainly rates as one of our favourite desktops. LXQt takes some of the aesthetic flourishes of KDE and other Qt apps and applies it to the LXDE framework – perfect for if you're the sort of person who uses a lot of Qt apps and would prefer a lightweight desktop, as it's the only lightweight Qt-based one around.

Recent benchmarks show RAM usage around 96MB, which puts it slightly higher than XFCE and significantly above LXDE. However, the developers of LXDE admit that with some better optimisation this number will surely go down and will definitely be lower than a GTK+ 3 XFCE.

The workflow is basically the same as LXDE, so moving over from one to the other isn't a big hassle.

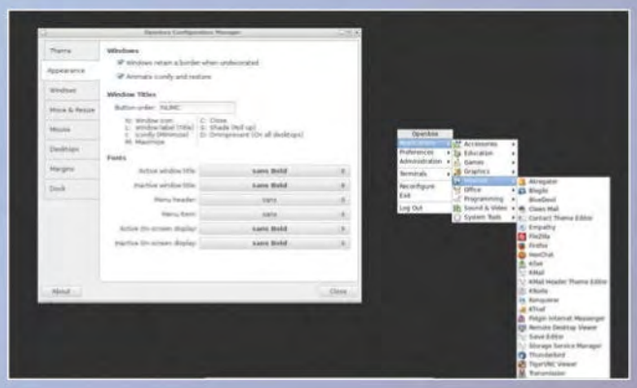
Best for
Old computers, test machines

Associated distros
None

www.lxqt.org

Openbox

The window manager can easily be used as a hyper-minimalist desktop



Above Openbox is really just a window manager and its minimalist design shows when used as a desktop

Pros
The lightest way to use a graphical interface and can be used on its own

Cons
Sacrifices user friendly design in favour of resources making it impractical

Openbox is usually thought of as more of a window manager – the thing that controls the window's aesthetics and placement within the desktop environment. However, Openbox can also be used entirely on its own without any other desktop overlay for the most minimal desktop imaginable, using the lowest amount of resources possible.

There are no panels by default, although if you want to then you can add some with other software, and clicking on the desktop opens a program menu that lets you access the different apps and some of the settings. However, Openbox can be not only quite impractical but difficult to get used to if you don't add any extra UI elements.

If you've installed LXDE then it will usually show up as an alternate desktop environment on login and is the main window manager of the lightweight desktop.

Best for
Developer systems

Associated distros
CrunchBang Linux, Lubuntu

www.openbox.org

CODE GNOME SOFTWARE

Discover the power and ease of **Vala** programming

C++'s success is obvious, but object-oriented programming can also be achieved in C. Developers were quick to capitalise on its success and construct design patterns based on the structures, function pointers and other similar niceties.

Vala is a C-based language that formalises these design patterns into language features in order to simplify their application. Its compiler transforms Vala code into normal C code that is then compiled using the system's native compiler. In theory, Vala code can consequently be run on any C-based operating system – this statement, of course, ignores the availability of libraries.

Compatibility is only one of the selling points here. Vala's development was guided by the intention to create a language that was tied into GTK and its underlying GObject model – this was a difficult set of paradigms that required significant effort from C programmers. Even though GTK powers a large array of programs, such as GIMP, its design patterns are unappealing to developers who are used to working with object-oriented GUI stacks. Developers working with the GTK framework will quickly see the benefits of Vala, as it provides a Qt-like experience for supported libraries and eliminates the 'repeat function invocation' design pattern. In short: creating a great-looking GTK app has never been so easy.

Finally, Vala's success has been validated by projects such as Elementary OS, which uses the programming language as its base language for a variety of components. Now that Vala is being utilised for such large projects it demonstrates the robustness of the underlying ecosystem.

While this article will focus on the technical aspects of Vala, let us start with the following, more general, example:

DEPLOY ON UBUNTU

On Ubuntu, the latest version of Vala can be downloaded via **apt-get** if the repository list is first expanded. Run:

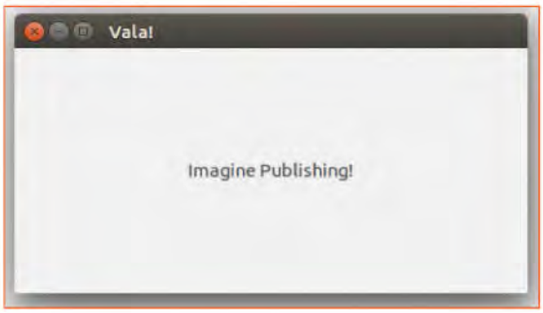
```
sudo add-apt-repository ppa:vala-team/ppa
sudo apt-get update
```

Due to compatibility reasons, the default vala and libgee packages provide outdated versions. Visit **bit.ly/1Fl0V69** in order to find the names of the most current editions; at the time of writing, the correct install command is:

```
sudo apt-get install libgee-0.8 vala-0.30 valadoc
```

Libgee is a package that provides easy access to a variety of storage classes. The Vala language specification intentionally excluded storage classes and similar trinkets for simplicity – this is because it would be pointless to reinvent them as they can already be found in GObject. Deploying our GUI-based examples on an Ubuntu system also requires the installation of the GTK+ GUI stack and its development support files:

```
sudo apt-get install libgtk-3-dev
```




```
using Gtk;

int main(string[] args)
{
 Gtk.init (ref args);

    var window = new Window ();
    window.title = "Vala!";
    window.border_width = 10;
    window.window_position = WindowPosition.CENTER;
    window.set_default_size (400, 200);
    window.destroy.connect (Gtk.main_quit);

    var l = new Label ("Imagine Publishing!");

    window.add (l);
    window.show_all ();

    Gtk.main ();
    return 0;

}
```

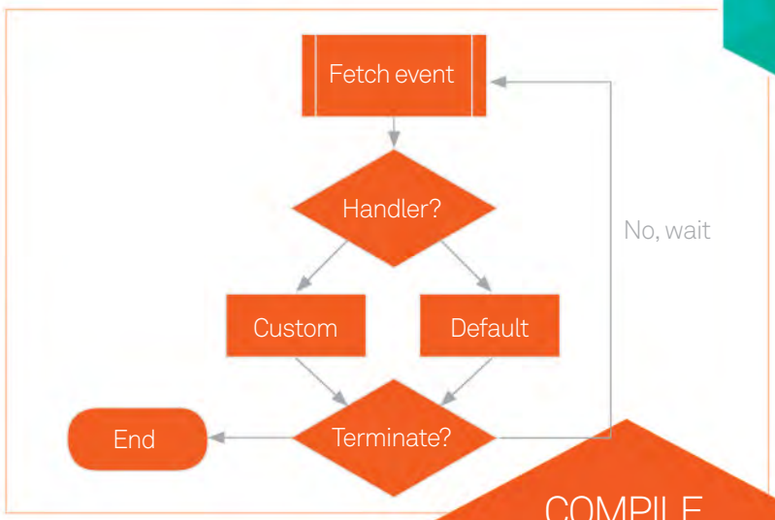
Even though this does not produce a particularly impressive form (as shown in the image above), it nevertheless illustrates a few basic concepts of Vala. First of all, namespaces are imported with the **using** command. **Gtk.init** is then invoked in order to begin the GTK application class.

Next, the **new** operator is used to create an instance of the Window class. It is responsible for creating a new form that can then be populated with widgets, such as the label instance stored. Finally, the event loop is started by calling **Gtk.main()**. From that moment onward, user and operating system events will be dispatched by an event loop. This structure is fundamental to most GUI toolkits – see the flowchart above for a basic overview.

Closing your application

Developers who start with Java's AWT GUI stack tend to create applications that can't be closed by clicking the usual X button. In GTK this can happen too, but our program avoids the problem by connecting the destroy signal to **Gtk.main_quit**.

Due to the importance of this design pattern, let us start by adding a button to the form at hand. The following code will show the changes that are needed in the file. It is also useful to know that Vala permits files containing multiple classes:




```
class HelperClass:Object{
    private Label myL;
    public HelperClass(Label _l)
    {
        myL=_l;
    }
    public void iWasClicked()
    {
        myL.set_text("Ow!");
    }
}

int main(string[] args){
 . . .
    var l = new Label ("Imagine Publishing!");
    var b = new Button.with_label ("Click me!");
    HelperClass myH=new HelperClass(l);
    b.clicked.connect(myH.iWasClicked);

    var box = new Box (Orientation.VERTICAL, 5);
    box.homogeneous = true;
    box.add (l);
    box.add (b);
    window.add (box);
    . . .
}
```

Using an event bus reduces cohesion between components: they no longer depend on one another, but only on the event bus and an event source.

The changes made to the GUI are fairly small: a box control was added to arrange the button and label below one another. **HelperClass** proves more interesting as its constructor takes a reference to the label that is then stored in a member variable. A second parameter points to the button, whose clicked signal is then connected to an event handler. Clicking the button will lead to a change in the contents of the label.

COMPILE AND EXECUTE

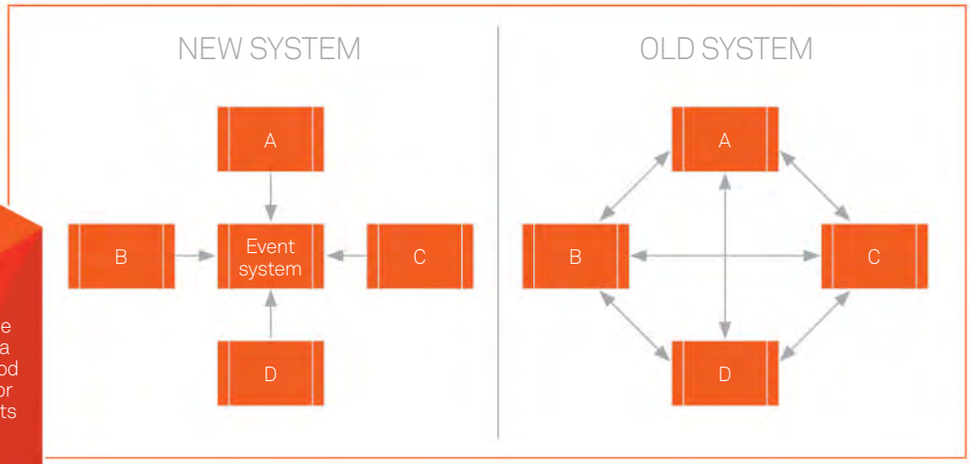
Vala's compiler goes by the name valac. When building with external packages, the **--pkg** parameter must be used in order to provide the context. Actual program execution can then be accomplished by running the result of the compilation process:

```
tam@han:~/valadir$ valac --pkg gtk+-3.0
vala-example1.vala
tam@han:~/valadir$ ./vala-example1
```

Note that the terminal cannot be closed until you are done with the program, as closing it prematurely shuts down the app too.

Top-left Our first Vala form – a simple window showing a bit of text

Top-right An event loop, for handling user and operating system events



Right Each component, rather than talking to every other one, only talks to the event bus

PARAMETERS AND POINTERS

Parameters can be left out at will – at least, as long as you start eliminating them from the right-hand side. Furthermore, Vala provides a convenience feature where the invoked method receives a pointer to the object responsible for the emission. The documentation illustrates its use via the following snippet:

```
public class Cl : Object {
public signal void some_event (int x, int y);
}
void on_some_event (Cl
source, int x, int y)
```

Vala provides a flexible set of signal definitions: a callback does not need to handle all of the parameters found in the emitted event. Test this by introducing an extra level of signal and callback routing. The corresponding HelperClass is:

```
class HelperClass:Object
{
    private Label myL;
    public signal void clickedSignal (int x, int y);
    public HelperClass(Label _l)
    {
        myl=_l;
        clickedSignal.connect(secondClicked);
    }
    public void secondClicked()
    {
        myL.set_text("Ow!");
    }
    public void iWasClicked()
    {
        clickedSignal(11,22);
    }
}
```

As before, we once again base our class on **Object**. Vala supports 'raw' classes not derived from Object, but this is a measure that is not recommended in most cases as Object comes with a large array of helpful methods.

Our new signal is declared as an 'empty' method decorated with the signal qualifier. The constructor connects it with **secondClicked**, while the original **iWasClicked** method demonstrates the signal invocation process.

Function invocations

An overloaded function means you can create two different versions of a function that will then take different parameters.

Since Vala code is intended to be usable from C (which does not know about overloading), developers are restricted to one set of parameters per method name. This limitation is softened slightly via the use of default parameters:

```
public void exampleFunction(int x, string st="Hello",
string st2="Oi")
{}
```

ExampleFunction can be invoked in the following three ways:

```
int main(string[] args)
{
    exampleFunction(1);
    exampleFunction(2, "Hi");
    exampleFunction(2, "Hi", "Bye");
```

Vala fills parameters via a left-to-right scheme. This means that the second invocation with one string will always populate **st**. You can't force the compiler to use the default for **st** while passing the provided value to **st2**. When working with constructors, a slightly odd design pattern should be used:

```
class HelperClass:Object
{
    public HelperClass(Label _l)
    {
        myl=_l;
        clickedSignal.connect(secondClicked);
    }
    public HelperClass.withInt(int i, Label _l)
    {
        myl=_l;
    }
```

Invocations can then be fully qualified. It could be accomplished via **=new HelperClass.withInt(...)** using the newly-created constructor. All Vala methods are final, which means they can't be overridden in inherited classes. Methods intended for overriding must use the following approach:

```
void main () {
```

```
    string boerte = "RIP, Börte!";
    string tank = "AGALAPANZER";
    for(int i=0;i<boerte.length;i++)
    {
        uint8 val=boerte[i];
        stdout.printf("Pos %i contains %c \n", i, val);
    }
    for(int i=0;i<tank.length;i++)
    {
        uint8 val=tank[i];
        stdout.printf("Pos %i contains %c \n", i, val);
    }
}
```

Processing strings

C is known for its barebones approach to handling strings. This might have worked when Kernighan and Ritchie wrote *The C Programming Language*, but 8-bit character sets have since proved it insufficient for characters like ä, ö ,ü or é.

Vala introduces a string data type. Strings are immutable: if you try to link or change them, a new one is allocated. Furthermore, the use of the UTF8 encoding scheme is mandatory. Creating a new string is easy:

```
string text = "This is a string that can be displayed";
```

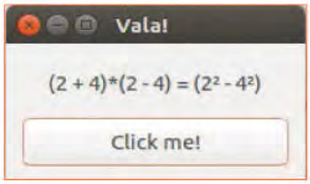
In practice, strings are often used to output one or more variables in a clear format. Vala addresses this with string templates, a feature best illustrated with an example:

```
int main(string[] args)
{
 Gtk.init (ref args);
    float av = 2;
    float bv = 4;
    string text = @"($av + $bv)*($av - $bv) = ($av² - $bv²)";
```

Prefixing a string with the **@** operator makes the runtime hunt for substrings prefixed with a **$**. They are then treated as calculation instructions. The example above leads to the result in the image shown below.

This is not limited to simple variable substitutions. The following performs a computation by performing a multiplication between the two values instructed:

```
int a = 7, b = 7;
string s = @"$a * $b = $(a * b)";   // => "7 * 7 = 49"
```



Left Variables prefixed with **$** are inserted in the string; brackets prefixed with **$** perform their calculations first

Keep in mind that the content of the template cannot be modified at runtime because **@** does not play nicely with string variables. This means that internationalisation requires the use of a selection statement or preprocessor directive. Let us start with a solution where the language can be chosen at runtime:

```
int main(string[] args)
{
 Gtk.init (ref args);
    float av = 2;
    float bv = 4;
    int lang=0;
    string text;
    if(lang==0)
    {
        text = @"AV ist $av, BV ist $bv";
    }
    else
    {
        text = @"AV is $av, BV is $bv";
    }
```

Preprocessor-based solutions are superior in that they reduce the size of the binary, as unneeded code is simply stripped out during the compile process. This means each language is to be realised via an executable file of its own:

```
int main(string[] args){
 Gtk.init (ref args);
    float av = 2;
    float bv = 4;
    string text;
    #if german
        text = @"AV ist $av, BV ist $bv";
    #elif english
        text = @"AV is $av, BV is $bv";
    #else
```

"Strings are immutable: if you try to link or change them, a new one is allocated"

ADVANTAGES OF GLIB.OBJECT

Even though Vala permits the creation of classes not derived from GLib.Object, most developers will be better served by foregoing the small amounts of memory saved in exchange for the more advanced features that are offered.

Most importantly, Glib.Object provides a reference-counting rubbish collector that permits Vala to dispose of unneeded object instances automatically. Secondarily, the base class provides a framework for property change notification. This means that clients can register interest in particular variables. If a change is performed via the corresponding method, the callback is then invoked.

Finally, GObject is filled with a list of helper functions that will simplify object construction, signal registration and signal management. Those of you who are friends of introspection can also use a few dedicated methods.

If a part of your product behaves oddly, it is best to start by checking whether all of your classes are derived from Glib.Object, because many a language feature acts up when it is confronted with 'normal' classes.

THE GARBAGE COLLECTOR

When creating a linked list or a similar data structure, developers tend to create data structures that are similar to the one below:

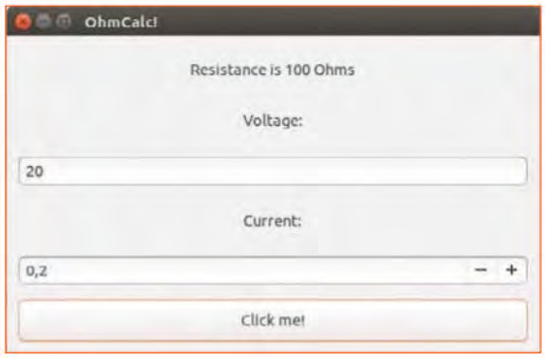
```
class AnEvilObject:GLib.Object
{
        public AnEvilObject prev;
        public AnEvilObject next;
}
```

While C# doesn't have any problem with this object, Vala is not able to destroy two such objects that point to one another. This is caused by the reference counting principle: each of the objects has an 'owner' and can consequently not be removed. The example demonstrates this behaviour. Run it on a workstation of your choice for a closer look:

```
public static int main(string[] args)
{
        while(1==1)
        {
                AnEvilObject obA=new AnEvilObject();
                AnEvilObject obB=new AnEvilObject();
                obA.next=obB;
                obB.prev=obA;
        }
        return 0;
}
```

Fortunately, an easy solution exists. Declaring one of the two references as 'weak' informs the rubbish collector that this particular relationship is not relevant for the determination of orphandom. The following version of AnEvilObject will subsequently work much better:

```
class AnEvilObject:GLib.Object
{
        public AnEvilObject prev;
        public weak AnEvilObject  next;
}
```



Above Here's the resistance calculator that we finish this article with shown in action

```
        oioioioi!!!!
#endif
```

Vala differs from C as **#define** is not supported; preprocessor commands can only be set from the command line using the syntax **valac --pkg gtk+-3.0 vala-example2.vala -D german**.

Another nasty trap involves the use of the [] operator. In C, [] can be used to traverse an array byte by byte: since ASCII strings have one byte per character, developers have come to expect a one-to-one correlation between position and character. Sadly, this is not the case due to UTF8 permitting multi-byte characters. As an example, let us use the AGALAPANZER code from earlier (across pages 170-171).

It runs over the entire length of the string, outputting each character to the command line. While AGALAPANZER – the name is mock German and proves helpful here when debugging, due to the A*A*A*A pattern – comes out in 11 bytes, the last salute looks odd. This is caused by the 'ö' in the name; UTF8 expresses it as a sequence of two bytes. The text below shows the output of the example program, while the following shows what is actually happening in memory.

```
Logic          R I P , B Ö R%T E !
In mem         R I P , B Ö%Ö R T E !
```

This shows the representation of the two strings as a character sequence (logic) and in memory (in mem). When looking at BÖRTE, the character Ö is split into two characters, breaking the alignment between characters and memory locations. Furthermore, be aware that [] is a read-only affair due to the immutability of the string class.

There are two more helpful elements in the string class. First, multi-line strings can be constructed as follows:

```
string verbatim = """This "verbatim string" can
contain escape sequences, such as \n, \t, \\, etc,
which are not processed.
Verbatim strings can also contain quotes and tend to
span multiple lines.""";
```

Secondarily, the == operator has grown in Vala. Its comparison power is not limited to traditional instance comparisons, instead it dives right into string data. This can be proven with the following:

```
string tA="Hello";
string tB="Hello";
if(tA==tB)
{
    text="Strings are equal";
}
```

String contains a few dozen helper functions that we don't have space here to discuss, but **valadoc.org/#!api=glib-2.0/string** is a great starting point.

Inheritance and interfaces

Like C#, Vala does not support multiple inheritance. Since some situations call for more complex hereditary hierarchies, Vala implements interfaces:

```
public interface IRelaxJuice : GLib.Object {
    public void sayManufacturerName()
    {
        stdout.printf("Relax Juice\n");
    }
    public abstract int myEAN();
}
class RelaxPeachJuice:IRelaxJuice, GLib.Object{
    public int myEAN(){
        return 2222;
    }
}
```

```
void main () {
    IRelaxJuice myIF=new RelaxPeachJuice();
}
```

A careful look at the implementation shown above reveals that Vala differs from C# in that interfaces can also contain methods. This is useful as it permits the 'oursourcing' of common logic: if an interface is to add a method, it does not need to be written over and over again in each of the child classes.

One more helpful feature enables interfaces to enforce the presence of a particular inheritance. As can be seen in the C# tutorial later in this issue, our IRelaxJuice can only be applied to objects that inherit from GLib.Object.

Generic classes

Being based on C, Vala is a statically typed language. This means that the data type of each and every expression must be known at compile time. JavaScript-trained programmers consider this burdensome, while most other coders praise the improved resilience against coding mistakes.

While typing something like **int** is hardly considered a big hassle, laying down the law on generics-based classes can be cumbersome. Let's look at the following statement:

```
MyGen1<string, MyBar<string, int>> elem = new
MyGen1<string, MyBar<string, int>>();
```

Var enables you to cut this down. Note that **var** is not a variant. In the example below, **elem** can't be filled with **int**, **string** or other classes that aren't compatible with the initial declaration:

```
var elem = new MyGen1<string, MyBar<string, int>>();
```

Let us finish by creating a simple generic class of our own and use it to create a resistance calculator program. The code that follows illustrates the basic structure and its invocation. Be aware that the similarities to C# and Java are intended.

```
using Gtk;

class OhmCalc:Object
{
private Label myL;
private Entry myU;
private SpinButton myI;

    public OhmCalc(Entry _u, SpinButton _i, Label _l)
    {
        myL=_l;
    myU=_u;
    myI=_i;
    }

    public void iWasClicked()
    {
double i=myI.get_value();
string uStore=myU.get_text();
double u=double.parse(uStore);
double res=u/i;
myL.set_text("Resistance is " +
res.to_string() + " Ohms");
    }
}

int main(string[] args)
{
 Gtk.init (ref args);
    var window = new Window ();

    window.title = "OhmCalc!";
    window.border_width = 10;
    window.window_position = WindowPosition.CENTER;
    window.set_default_size (500, 300);
    window.destroy.connect (Gtk.main_quit);

    var l = new Label ("Result");
    var u = new Gtk.Entry ();
    var i = new SpinButton.with_range (0,200,0.1);
    var b = new Button.with_label ("Click me!");
    var myOhmcalc=new OhmCalc(u,i,l);
    b.clicked.connect(myOhmcalc.iWasClicked);

    var box = new Box (Orientation.VERTICAL, 5);
    box.homogeneous = true;
    box.add (l);
box.add (new Label ("Voltage:"));
    box.add (u);
box.add (new Label ("Current:"));
    box.add (i);
    box.add (b);
    window.add (box);
    window.show_all ();

    Gtk.main ();
    return 0;
}
```

OIOIOIOI!!!! – THAT'S INVALID

The 'oioioi' line we used earlier is a classic preprocessor trick used by experienced hands. If a coder forgets to pick a valid choice, the preprocessor will present the compiler with 'oioioioi!!!!' This nice expression is invalid Vala, thereby leading to the aborting of the whole compilation process.

The designers working on new programming languages must strike the difficult balance between research results and established design patterns.

The Vala team have accomplished some amazing work because the language remains similar to C and C++, while still enabling its users to participate in advances in the area of language design. Moreover, the first class integration into the GUI stack makes creating applications a breeze, while the C interface permits you to reuse existing code.

If you find yourself working on GTK-based code and developing software for GNOME, then deploying Vala really is a must. ■

"Interfaces can also contain methods. This permits the 'oursourcing' of common logic"

About the mag

Dedicated to all things Linux

Written for you

Linux User & Developer is the only magazine dedicated to advanced users, developers & IT professionals

In-depth guides & features

Written by grass-roots developers and industry experts

Free assets every issue

Four of the hottest distros feature every month – log in to FileSilo, download and test them all!

subscribers to. . .

LinuxUser & Developer™

Try 3 issues for £5 in the UK*
or just $7.85 per issue in the USA**
(saving 54% off the newsstand price)

For amazing offers please visit
www.imaginesubs.co.uk/lud

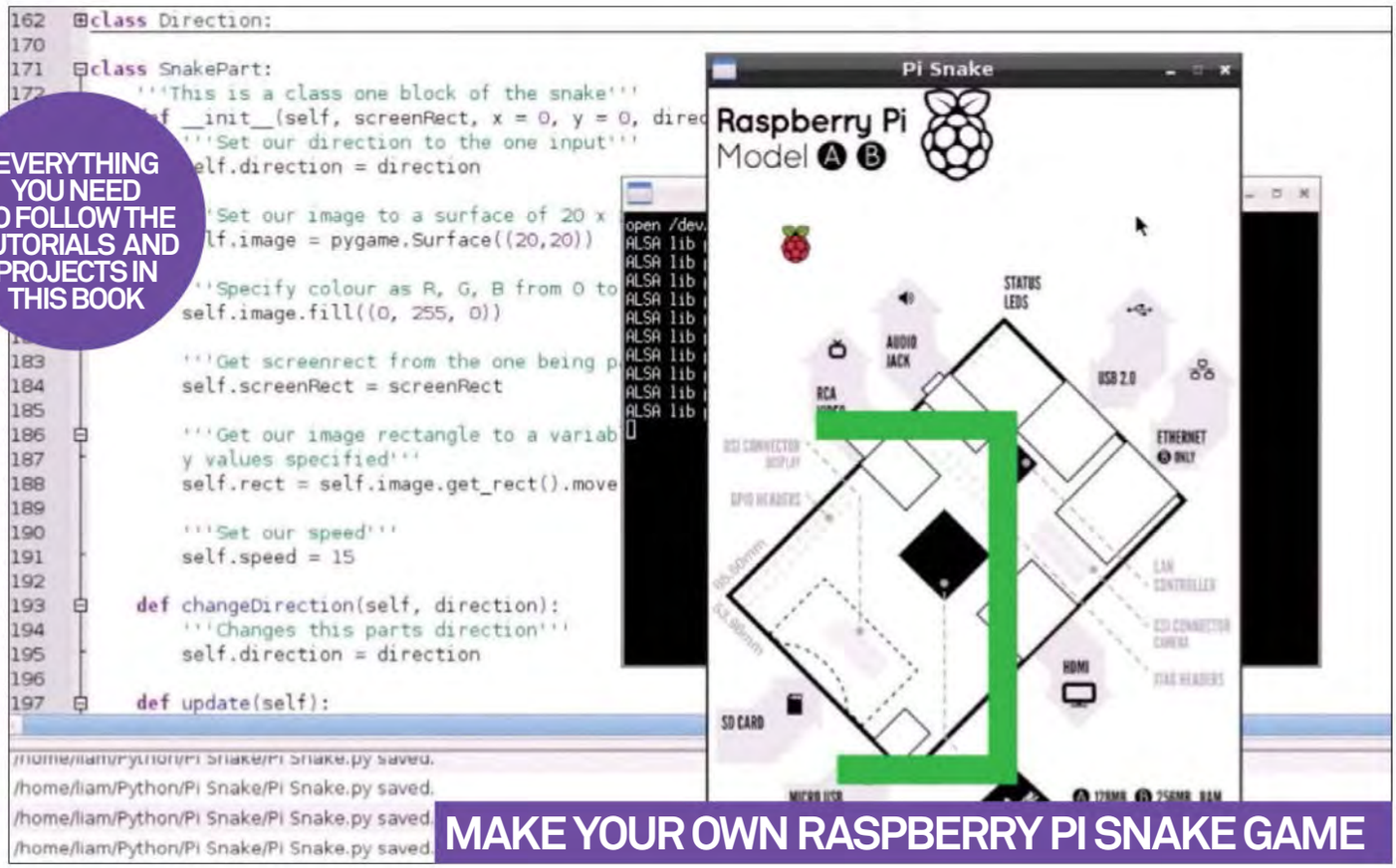
Quote code ZGGZINE

Or telephone UK 0844 249 0282+ overseas +44 (0) 1795 418 661

+ Calls will cost 7p per minute plus your telephone company's access charge

YOUR FREE RESOURCES

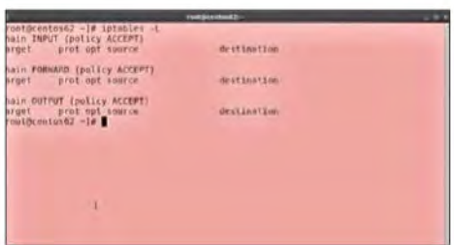
Log in to **filesilo.co.uk/bks-813/** and download your tutorial resources **NOW!**



EVERYTHING YOU NEED TO FOLLOW THE TUTORIALS AND PROJECTS IN THIS BOOK

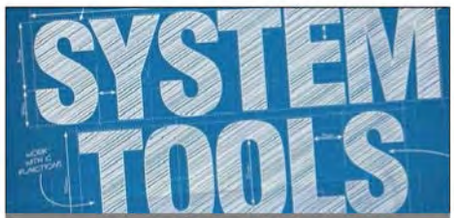
MAKE YOUR OWN RASPBERRY PI SNAKE GAME

All the files you need to complete the tutorials



2+ hours of video tutorials

PACKED WITH FREE PREMIUM CONTENT



Develop new Linux skills

YOUR BONUS RESOURCES

ON FILESILO WE'VE PROVIDED FREE, EXCLUSIVE CONTENT FOR **LINUX & OPEN SOURCE ANNUAL** READERS, INCLUDING...

- 31 free video tutorials to help enhance your Linux skills, from mastering Debian, webinars on Android in embedded systems, OpenStack, Unix and more, to a full guide to making your own Raspberry Pi game.
- All of the tutorial files you need to complete the projects in the book including programming a Parrot AR drone and using games controllers with Linux.
- Code listings for projects and features in the book to help improve your systems.

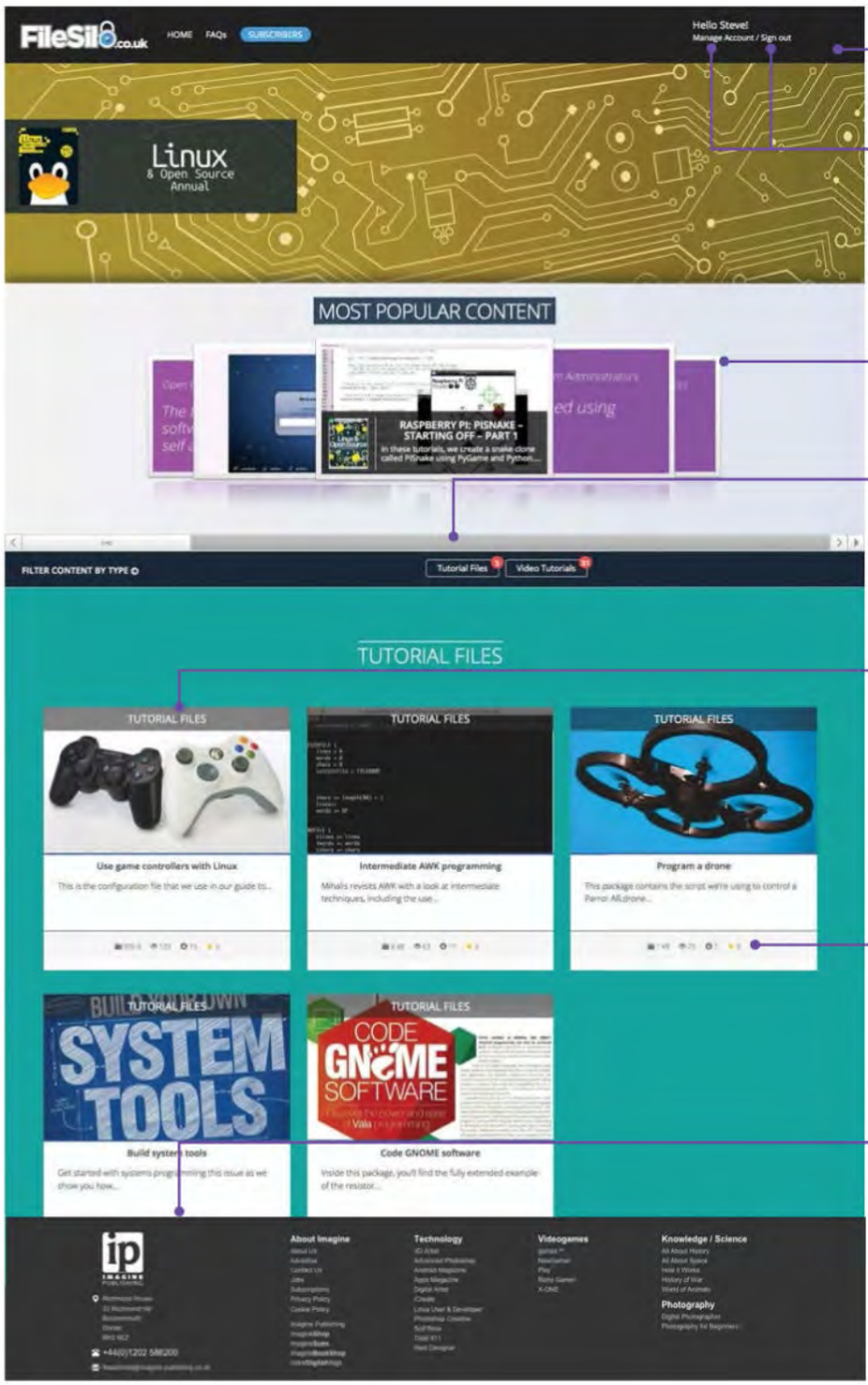
FileSilo

Go to: http://www.filesilo.co.uk/bks-813/

FILESILO – THE HOME OF PRO RESOURCES

Discover your free online assets

- A rapidly growing library
- Updated continually with cool resources
- Lets you keep your downloads organised
- Browse and access your content from anywhere
- No more torn disc pages to ruin your magazines
- No more broken discs
- Print subscribers get all the content
- Digital magazine owners get all the content too!
- Each issue's content is free with your magazine
- Secure online access to your free resources



This is the new FileSilo site that replaces your disc. You'll find it by visiting the link on the following page.

The first time you use FileSilo, you'll need to register. After that, you can use your email address and password to log in.

The most popular downloads are shown in the carousel here, so check out what your fellow readers are enjoying.

If you're looking for a particular type of content, like software or video tutorials, use the filters here to refine your search.

Whether it's tutorials or software resources, categories make it easy to identify the content you're looking for.

See key details for each resource including number of views and downloads, and the community rating.

Find out more about our online stores, and useful FAQs, such as our cookie and privacy policies and contact details.

Discover our fantastic sister magazines and the wealth of content and information that they provide.

HOW TO USE FileSilo

EVERYTHING YOU NEED TO KNOW ABOUT ACCESSING YOUR NEW DIGITAL REPOSITORY

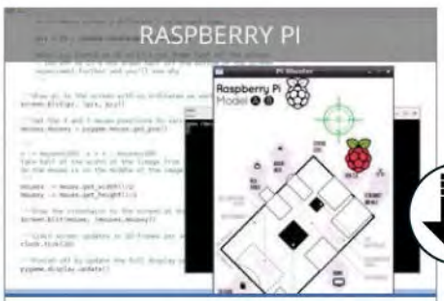
To access FileSilo, please visit http://www.filesilo.co.uk/bks-813/

01 Follow the on-screen instructions to create an account with our secure FileSilo system, or log in and unlock the issue by answering a simple question about the edition you've just read. You can access the content for free with each edition released.

02 Once you have logged in, you are free to explore the wealth of content made available for free on FileSilo, from great video tutorials and online guides to superb downloadable resources. And the more bookazines you purchase, the more your instantly accessible collection of digital content will grow.

03 You can access FileSilo on any desktop, tablet or smartphone device using any popular browser (such as Safari, Firefox or Google Chrome). However, we recommend that you use a desktop to download content, as you may not be able to download files to your phone or tablet.

04 If you have any problems with accessing content on FileSilo, or with the registration process, take a look at the FAQs online or email filesilohelp@imagine-publishing.co.uk.



PiSnake – Starting off – Part 1

In these tutorials, we create a snake clone called PiSnake using PyGame...

Intro to Debian Linux

In this video, Liam gives us an insight into working with Debian...

Code GNOME software

Inside this package, you'll find the fully extended example of the resistor...

NEED HELP WITH THE TUTORIALS?

Having trouble with any of the techniques in this issue's tutorials? Don't know how to make the best use of your free resources? Want to have your work critiqued by those in the know? Then why not visit the Bookazines or Linux User & Developer Facebook page for all your questions, concerns and qualms. There is a friendly community of experts to help you out, as well as regular posts and updates from the bookazine team. Like us today and start chatting!

facebook.com/ImagineBookazines
facebook.com/LinuxUserUK

Linux & Open Source Annual

Everything you need to master open source software and operating systems

Over 2 hours of free video tutorials

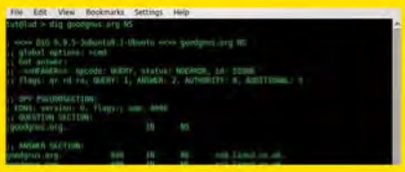
Open source world

Get to know Ubuntu phone, discover Plasma mobile and see the open-source future

Enhance your system

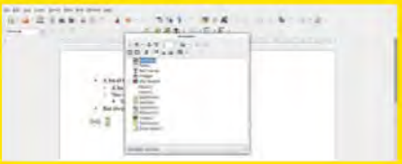
Build system tools to switch up Linux and troubleshoot network issues like a pro

Programming in Linux



Master the command line with 100 tricks, learn all about C# and automate your home

Harness FOSS



Find out how to virtualise Linux and discover 20 essential LibreOffice tips